Universität Hamburg

Fakultät für Mathematik,
Informatik und Naturwissenschaften

Department Informatik

Zentrum für Verteilte Informations-
und Kommunikationssysteme

Arbeitsbereich Verteilte Systeme
und Informationssysteme

Kontextbasierte Kooperation: Unterstützung verteilter Prozesse im Mobile Computing

Dissertation

zum Erlangen des Doktorgrades
an der Fakultät für Mathematik,
Informatik und Naturwissenschaften
der Universität Hamburg

Dipl.-Inform. Christian Philip Kunze

Betreuer und 1. Gutachter:	Prof. Dr. Winfried H. Lamersdorf, Universität Hamburg
2. Gutachter:	Prof. Dr. Stefan Fischer, Universität zu Lübeck
Tag der Disputation:	25. Juni 2008

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein, 1879 – 1955

Zusammenfassung

In der modernen Dienstleistungsgesellschaft ist *Mobilität* - z.B. von Personen, Waren, Daten oder Diensten - zu einem wichtigen Faktor geworden. In Kombination mit dem derzeit in der Informations- und Kommunikationstechnologie vorherrschenden Trend zur Dezentralisierung sind daher inzwischen eine Vielzahl von portablen und miteinander vernetzten Geräten entstanden, deren Einsatz die *ubiquitäre Verfügbarkeit von Informationen und Dienstleistungen* für mobile Benutzer kontinuierlich gesteigert hat. Dazu sind - neben der Portabilität von Geräten - aber auch Softwaresysteme mobil geworden, indem sie auf mobilen Geräten ablaufen können, sich der Mobilität dieser Geräte bewusst werden und sich so möglichst optimal an die sich dynamisch verändernden Ausführungskontexte anpassen können.

Vor diesem Hintergrund hat das Forschungsgebiet des *Mobile Computing* zum Ziel, moderne mobile Informationssysteme durch innovative Systemsoftware, Middleware-Systeme und Anwendungen an die sich stetig weiter entwickelnden Bedürfnisse der Benutzer anzupassen. Dabei nehmen *kontextbezogene Systeme* eine zentrale Rolle ein, da sie durch die Integration von Kontextwissen mobile Applikationen flexibel und anpassungsfähig machen. Jedoch sind solche Systeme bisher meist noch eher ad-hoc statisch, monolithisch strukturiert sowie geschlossen und daher insgesamt für spezielle und eher kurzzeitige Aufgaben konzipiert. Mobile Systeme, die sich auch an den langfristigen und oftmals spontanen Aufgaben und Anforderungen der Benutzer orientieren, sind mit den etablierten Verfahren und Methoden daher bislang kaum realiserbar.

Die vorliegende Arbeit untersucht daher die Defizite aktueller mobiler Systeme sowie der sie unterstützenden Middleware-Plattformen. Basierend auf den Ergebnissen dieser Untersuchung wird das Konzept der *kontextbasierten Kooperation* erarbeitet, das eine Unterstützung von benutzerzentrischen, langlebigen und komplexen Aufgaben in mobilen Systemumgebungen als konsequente Weiterentwicklung etablierter kontextbezogener Konzepte ermöglicht. Durch das Zusammenführen der kontextbasierten Kooperation mit den Konzepten und Methoden der *Service-Oriented Architecture* wird ein entsprechendes Umsetzungskonzept in Form von *Mobilen Prozessen* erarbeitet und eingeführt. Zudem wird mittels einer prototypischen Implementierung einer Systemplattform für Mobile Prozesse die praktische Realisierbarkeit der eingeführten Konzepte belegt. Mit Hilfe dieser Plattform werden ferner die Konzepte der kontextbasierten Kooperation und der Mobilen Prozesse evaluiert und bewertet.

Als Ergebnis dieser Arbeit wurde eine *Middleware-Systemplattform* erarbeitet, die es (mobilen) Umgebungen ermöglicht, ad-hoc entstehende, komplexe Prozesse ohne zentrale Koordinierungsinstanz auszuführen. Somit stellt der eingeführte Ansatz eine wesentliche Weiterentwicklung kontextbezogener Systeme im Sinn des Ubiquitous Computing dar.

Abstract

Within the modern services society, *mobility* has become one of the main driving factors. Supported by the trend of decentralization in information and communication technology, a great diversity of portable and networked devices has evolved. The use of these devices has increased continuously and leads to a *ubiquitous availability of information and services* for mobile users. Next to the portability of devices, software systems themselves have become mobile - enabled by an increased awareness of the devices' mobility and thus by the ability to adjust to highly changing execution contexts.

With this background, the research field of *Mobile Computing* subsumes efforts to adapt modern mobile information systems, middleware platforms, and applications to new evolving needs of mobile users. In doing so, *context-based systems* play a major role, as these systems supplement mobile applications by adding context knowledge to achieve flexibility and adaptability for such systems. However, in reality most of these currently existing systems are rather ad-hoc static, monolithic structured, and closed. In consequence they are mostly restricted to support rather specific and short time tasks. On the other hand, more general systems being able to support also long running and spontaneous user tasks, are still hardly realizable with existing methods and techniques.

Therefore, this thesis adresses these shortcomings of current mobile systems and supporting middleware platforms in a detailed study. On the basis of the results of this study, the concept of *context-based cooperation* is proposed and elaborated in detail. As a consequent evolution of concepts derived from established context-based systems, this approach serves to integrate support of user-centric, long running, and ad-hoc tasks into (new) mobile system environments. Finally, by merging context-based cooperation with well-established concepts and methods of *Service-Oriented Software Architectures*, the realization concept of *Mobile Processes* is elaborated. Based on that, a prototype implementation of a system platform for Mobile Processes is realized in order to prove the practical realizability as well as to evaluate the feasibility of the introduced concepts of context-based cooperation and Mobile Processes.

Finally, the practical result of this thesis is a *middleware system platform* that enables (mobile) environments to execute ad-hoc arising, complex processes without a central coordination mechanism. Thus, the introduced approach represents an important enhancement of context-based systems towards the vision of Ubiquitous Computing.

Danksagung

Bei der Durchführung meines Dissertationsprojekts haben mich viele Menschen unterstützt, denen ich an dieser Stelle besonderen Dank sagen möchte:
Zunächst möchte ich mich bei meinem Betreuer Herrn Professor Winfried Lamersdorf bedanken, der mich seit dem Studium auf meinem akademischen Weg begleitet und mir als wissenschaftlichem Mitarbeiter an seinem Arbeitsbereich diese Dissertation ermöglicht hat.
Des Weiteren danke ich meinen Kollegen vom Arbeitsbereich Verteilte Systeme und Informationssysteme, die mir durch ihren fachlichen Rat und ihre moralische Unterstützung bei der Erstellung dieser Arbeit geholfen haben. Im Besonderen danke ich Frau Sonja Zaplata, Frau Anne Awizen und Herrn Volker Nötzold, die stets für fachlichen, technischen und zwischenmenschlichen Rückhalt gesorgt haben.
Ein entscheidender Faktor für das Gelingen dieser Arbeit ist auch die Zusammenarbeit mit den Studenten gewesen, die sich in meinem Promotionsprojekt DEMAC engagiert haben. Deshalb möchte ich Herrn Heiko Adam, Herrn Victor Dreiling, Herrn Alexander Holbreich, Herrn Mirwais Turjalei, Herrn Ante Vilenica und Frau Alice Winnicki meinen besonderen Dank ausdrücken.
Sehr am Herzen liegt mir jedoch der Dank an meine Familie und meine Freunde, die mir stets den Rücken frei gehalten und mich über so manche Unwägbarkeit hinweggetragen haben. Besonders möchte ich mich dabei bei meiner Mutter Frau Dr. Anke Kunze bedanken, die durch ihr stetiges Engagement und ihre Motivation maßgeblich dazu beigetragen hat, dass ich diese Dissertation anfertigen konnte. Zudem möchte ich Frau Martina Bünsow, Frau Claudia Fischer und Frau Maud Schwarz für ihre Qualitätssicherung danken.

Christian P. Kunze

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In der aktuellen Entwicklung der Informatik verstärkt sich der Trend zur *Dezentralisierung* der eingesetzten *Informationstechnologie* sowohl im Bereich der Hard- als auch der Software. Dabei dringt die Computertechnologie immer weiter in den Alltag der Menschen ein und unterstützt diese bei einer Vielzahl von unterschiedlichen privaten sowie geschäftlichen Aktivitäten – und das nahezu an jedem Ort und zu jeder Zeit [HMNS03]. Diese Entwicklung wird vor allem durch die stetigen *Fortschritte in der Informations- und Kommunikationstechnologie* (IuK) vorangetrieben, wobei die dabei eingesetzten Geräte nicht nur immer leistungsfähiger, sondern auch immer kleiner, kostengünstiger und mobiler werden. Waren sie zu Beginn dieser Entwicklung zum Teil noch raumfüllend und nur für wenige erschwinglich, so haben heutzutage Laptops und andere mobile Endgeräte eine große und noch immer überproportional wachsende Verbreitung in den weltweiten Haushalten gefunden (vgl. Abbildung 1.1). Daneben befinden sich Mikroprozessoren aber auch in vielen Alltagsgegenständen wie Haushaltsgeräten, Fahrzeugen, Mobiltelefonen oder persönlichen digitalen Assistenten (PDA), die zudem immer häufiger auch untereinander und mit dem Internet verbunden sind. Damit steigt neben der Dezentralisierung der Informationstechnologie auch die ubiquitäre Verfügbarkeit von Informationen und Dienstleistungen stetig an [Mat05a].

Vor dem Hintergrund dieser Entwicklung einer immer größer werdenden Penetration moderner IuK-Technologie in den alltäglichen Raum ist die Vision des *Ubiquitous Computing* entstanden. Hierbei treten Computer (und andere elektronische Geräte) immer mehr in den Hintergrund und deren Benutzer erhalten eine – oft unbewusste – Unterstützung durch vernetzte Geräte und deren Kollaboration bei beliebigen Aufgaben an beliebigen Orten [Wei91, Wei93]. Dass in einem solchen Szenario die Mobilität in ihren verschiedenen Ausprägungen eine zentrale Rolle spielt, ist geradezu offensichtlich: Mobile *Benutzer* bewegen sich von Ort zu Ort und benötigen dabei – möglichst zu jeder Zeit und von jedem Platz aus – Zugriff auf ihre Anwendungen und Daten. Aber auch die *Geräte* selbst können mobil sein und so dynamische Systeme bilden, in denen Daten und Dienstleistungen untereinander ausgetauscht werden. In solchen Umgebungen können sich sogar Teile von *Anwendungen* als Einheiten migrierenden Co-

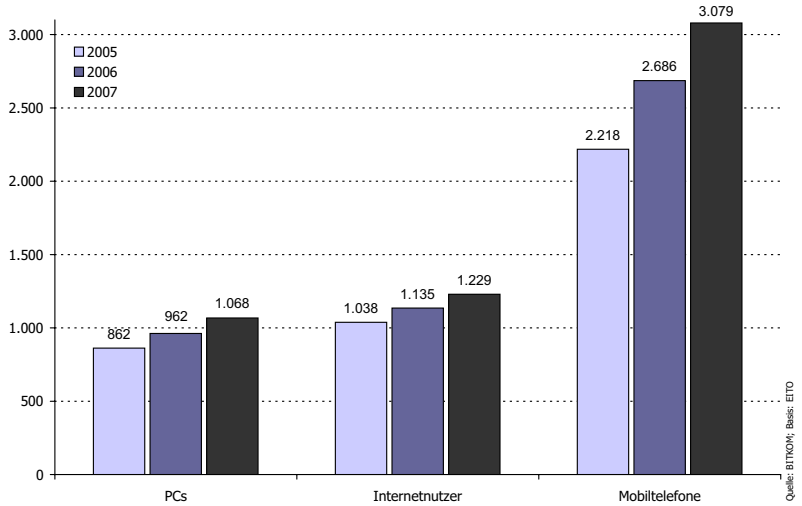


Abbildung 1.1.: Entwicklung weltweiter Informationsinfrastrukturen (in Millionen / nach (Bun07))

des zwischen verschiedenen Geräten bewegen, um dort weiterverarbeitet oder ausgeführt zu werden [Kun05a].

Um allerdings solche Aspekte der Mobilität auf Systemebene zu unterstützen, sind zusätzliche Anforderungen und Einschränkungen im Verhältnis zu klassischen Verteilten Systemen zu beachten. Vor allem hat sich gezeigt, dass das (in klassischen Verteilten Systemen meist vorherrschende) Streben nach möglichst weit gehender *Verteilungstransparenz* den mobilen Systemen nicht immer gerecht wird. Im Gegenteil, durch das dort typischerweise oft auftretende Ungleichgewicht von benötigten und angebotenen Ressourcen können mobile Anwendungen nur dann den Benutzer optimal unterstützen, wenn sie sich ihrer Mobilität bewusst werden (*Awareness*) und sie sich zudem auch den wechselnden Verhältnissen einer solchen mobilen Umgebung anpassen können (*Adaptability*) [Sat04].

Sowohl im Ubiquitous Computing als auch bei der Unterstützung mobiler Systeme muss somit der Benutzer mit seinen Aufgaben und Bedürfnissen noch mehr in den Mittelpunkt des Systementwurfs gestellt werden. Benutzer möglichst optimal und zugleich diskret beim Erledigen ihrer jeweiligen Aufgaben zu helfen, sollte also ein vorrangiges Ziel von Middleware-Plattformen im *Mobile Computing* sein. Dabei müssen solche Systemplattformen einen Rahmen bieten, in dem die Geräte einer mobilen Umgebung von Spezialisten zu Generalisten werden, um die meist a priori unbekannten und ad hoc entstehenden Aufgaben und Bedürfnisse eines (mobilen) Benutzers auch angemessen unterstützen zu können.

Dementsprechend ergründet die vorliegende Arbeit zunächst, wie die

immer umfangreicher werdenden Aufgaben und Anforderungen solcher Nutzer von mobilen Systemumgebungen möglichst optimal und so unaufdringlich wie möglich durch die Kooperation unterschiedlicher Geräte unterstützt u nd erfülltwerden können. Dazu wird auch auf die Frage eingegangen werden, wie diese benutzerzentrischen komplexen Aufgaben in einem abstrakten und generischen Modell für mobile Systeme beschrieben und kooperativ ausgeführt werden können. Auf Basis eines solchen Modells wird dann ein Integrationskonzept für eine entsprechende Middleware-Plattform herausgearbeitet, die das Beschreiben und Ausführen solcher benutzerzentrischer Applikationen in einer verteilten und mobilen Umgebung unterstützt.

1.1. Defizite aktueller Middleware-Plattformen für mobile Systeme

Durch die Dezentralisierung und ubiquitäre Verfügbarkeit von Informationen und Dienstleistungen nimmt das Forschungsgebiet des Mobile Computings eine zentrale Rolle bei der Realisierung moderner IuK-Infrastrukturen ein. Dabei müssen neben entsprechenden Hardware-Voraussetzungen auch Softwaresysteme entwickelt werden, die helfen (komplexe) Anwendungen in einer mobilen Umgebung zu realisieren. Solche unterstützenden Softwareinfrastrukturen zur Entwicklung und zum Betrieb von verteilten mobilen Anwendungen (*mobile Middleware-Plattformen*) können hierbei zum Teil etablierte Verfahren und Methoden traditioneller Verteilter Systeme nutzen, müssen aber aufgrund der hohen Dynamik und der oftmals bestehenden Ressourcenknappheit und Heterogenität der beteiligten Systeme auch erweiterte sowie individuelle Eigenschaften besitzen. Eine zentrale Herausforderung solcher Middleware-Plattformen ist es dabei, einen Ausgleich zwischen den zum Teil einander diametral entgegenstehenden Anforderungen mobiler Systeme herzustellen, um dennoch eine möglichst optimale Unterstützung für die Bedürfnisse und Anwendungen mobiler Anwender zu bieten [Sat96].

Bestehende Middleware-Systeme für Anwendungen des Mobile Computing spiegeln die evolutionären Entwicklungen und Erkenntnisse in diesem Teilgebiet wider: Ging es anfangs eher darum, traditionelle Middleware-Ansätze auf die meist unzuverlässigen und variierenden Kommunikationsbedingungen in mobilen Systemen durch die Verwendung asynchroner Kommunikationsparadigmen anzupassen [MCE02, SAS01], so hat sich inzwischen gezeigt, dass dies allein nicht genügt, um möglichst op-

timale Middleware Systeme für mobile Umgebungen zu schaffen. Denn asynchrone Kommunikationsparadigmen führen zwar zu einer zeitlichen Entkopplung von Sender und Empfänger einer Nachricht oder Information, jedoch wird damit der Heterogenität und Dynamik im Systemen mobiler Nutzer und Geräte nicht ausreichend Rechnung getragen. Im Allgemeinen besitzen solche Systeme damit zwar eine hohe Skalierbarkeit und Offenheit zur Integration heterogener Geräte, jedoch kann aufgrund der angestrebten Verteilungstransparenz sowie der beschränkten Ressourcen und Dienstleistungen, die in einem mobilen System vorherrschen, nur eine sehr rudimentäre Unterstützung mobiler verteilter Anwendungen erreicht werden. Aus dieser Beobachtung heraus hat sich die Erkenntnis entwickelt, dass mobile Systeme sich zudem ihrer Umgebung bewusst werden müssen und auf Basis dieses Kontextbewusstseins sich den verändernden Bedingungen anpassen sollten [MCE02, CK00, Dey00].

Solche durch das Etablieren eines Kontextbewusstseins erweiterten Systeme führen damit zu kontextbezogenen Anwendungen, die sich der Dynamik und Heterogenität mobiler Umgebungen anpassen können. Hierdurch ermöglichen entsprechende Middleware-Plattformen neuartige Anwendungsklassen, die den Charakteristika mobiler Systeme besser gerecht werden, indem sie Informationen und Daten kontextbezogen auswählen oder präsentieren, ihre Aktionen auf den Kontext abstimmen beziehungsweise selbst wieder Einfluss auf den Kontext nehmen. Damit werden der Kontext und seine Modellierung sowie der Aufbau und die Verwaltung des Kontexts jeweils wichtige Teile von Middleware-Systemen im Mobile Computing und den hierauf aufbauenden Anwendungen [Bec04].

Durch die bisher meist verwendeten asynchronen Kommunikationsparadigmen und den Kontextbezug adressieren die zurzeit etablierten Middleware-Systeme im Mobile Computing bereits wichtige Herausforderungen mobiler Systeme. Durch die Entwicklung von kommunikationsorientierten Middleware-Plattformen hin zu solchen mit Kontextbezug sind dabei Systemplattformen entstanden, die einen deutlichen Anwendungsbezug aufweisen. Diese Entwicklung geht einher mit einer engeren Verzahnung zwischen Middleware und Anwendung, da so die Ereignisse und Daten des Kontexts direkten und damit funktionalen Einfluss auf die Anwendung bekommen können. Jedoch führt die Verwendung solcher Middleware-Plattformen noch immer zu eher ad-hoc-statischen, monolithisch strukturierten und geschlossenen Anwendungen, die für spezielle und eher kurzzeitige Aufgaben konzipiert sind. Der hier logisch anschließende Schritt muss also ein Hinwenden zu Systemen sein, die auch langlebige und komplexe

Aufgaben unterstützen, die zum Teil erst spontan aus den Bedürfnissen und Aktionen des mobilen Benutzers entstehen. Dies führt auch dazu, dass neuartige Middleware-Plattformen im Mobile Computing eine deutliche Benutzerorientierung bieten müssen [KZL06, Kun05b].

1.2. Zielsetzung und Methodik der Arbeit

Vor dem Hintergrund der vorab dargelegten Defizite aktueller Middleware-Plattformen für mobile Systeme zielt die Aufgabenstellung der vorliegenden Arbeit auf die Konzeption und Realisierung eines erweiterten Middleware-Rahmenwerks zur Unterstützung benutzerzentrischer, langlebiger und komplexer Aufgaben in mobilen Systemumgebungen. Hierbei soll das dabei zu entwickelnde Konzept und dessen Realisierung eine konsequente Weiterentwicklung etablierter Systemansätze darstellen und durch kontextbezogene sowie kooperative Systemstrukturen die Potenziale einer mobilen Umgebung dem Nutzer möglichst umfassend zugänglich machen.

Konzeption kontextbezogener Kooperation in mobilen Umgebungen

Der konzeptionelle Anteil dieser Arbeit zielt auf die Erweiterung bestehender kontextbezogener Middleware-Konzepte im Mobile Computing im Sinne der oben genannten Anforderungen. Dabei liegt der Fokus auf der Etablierung und Realisierung kontextbezogener Kooperationsformen, um die Fähigkeiten einer mobilen Umgebung deren Nutzern in einem höheren Grad zugänglich zu machen als es bisher möglich ist. Um ein solches Konzept zu begründen, wird dazu in einem ersten Schritt herausgearbeitet, in welchem Umfang kooperative Interaktionsformen durch etablierte Konzepte im Mobile Computing bereits abgedeckt werden, um daraus die konzeptionelle Erweiterung abzuleiten. Hierbei sollen möglichst beliebig komplexe Aufgaben und Ziele der Nutzer einer mobilen Umgebung berücksichtigt werden. Ziel ist es dabei, auf diese Weise bisher nicht umsetzbare Anwendungen und Aufgaben auch mit ressourcenbegrenzten mobilen Systemen ausführen zu können. Entsprechend wird ein angepasster Kooperationsbegriff für das Mobile Computing eingeführt und als konzeptionelles Modell etabliert.

In einem zweiten Schritt wird dann durch die Verbindung des kontextbezogenen Kooperationsbegriffs mit etablierten Konzepten und Methoden der *Service-Oriented Architecture* (SOA) ein konzeptionelles Realisierungsmodell herausgearbeitet. Ziel ist es hier, das abstrakte Modell der kontextbasierten Kooperation in ein Konzept zu überführen, das die abstrakten

Vorgaben und Eigenschaften in einer heterogenen, unzuverlässigen und sich dynamisch ändernden Umgebung berücksichtigt. Hierbei müssen jedoch nicht nur die Bedürfnisse und Eigenschaften, die aus der mobilen Umgebung erwachsen, einfließen, sondern auch diejenigen Anforderungen, die sich aus einer benutzerzentrischen Sicht auf das System ergeben. Dies ist deshalb notwendig, da das Realisierungskonzept einen weiteren Schritt hin zur Umsetzung der Vision des *Ubiquitous Computing* beitragen soll, in der Nutzer unauffällig, aber effektiv durch ihre Umgebung bei der Durchführung ihrer jeweiligen Aufgaben unterstützt werden.

Realisierung kontextbasierter Kooperation als Mobile Prozesse

Der praktische Anteil der Arbeit stellt eine technisch angemessen Realisierung des auf der SOA basierenden Konzepts der kontextbasierten Kooperation in Form von migrierenden und verteilt ausgeführten Prozessen (*Mobile Prozesse*) dar. Dazu wird eine Middleware-Plattform für mobile Umgebungen vorgeschlagen und implementiert, die den Aufbau eines kontextbezogenen ad-hoc *Overlay-Netzwerks* zur Ausführung Mobiler Prozesse unterstützt. Für diese mobile Middleware-Plattform sind dabei sowohl ein genereller Entwurf als auch eine prototypische Implementierung zu erarbeiten.

Eine Middleware zur Realisierung Mobiler Prozesse muss sowohl ein geeignetes Prozessmodell zur abstrakten Beschreibung komplexer Aufgaben sowie ihrer nicht-funktionalen Rahmenbedingungen als auch ein abgestimmtes Kontext- und Kommunikationsmodell umfassen. Hierbei muss das Prozessmodell so konzipiert sein, dass es die Dynamik und Heterogenität mobiler Umgebungen berücksichtigt und auch unter unzuverlässigen Kommunikationsbedingungen die Ausführung von Prozessen ermöglicht. Um dies zu erreichen, muss für die einzelnen Teilaufgaben eines Mobilen Prozesses (*Aktivitäten*) eine abstrakte Beschreibungsform etabliert werden, die nicht nur von der einzelnen, diese Aktivität ausführenden Dienstinstanz abstrahiert, sondern zusätzlich auch von der zu verwendenden Implementierungstechnologie (*abstrakte Dienstklasse*). Dies ist deshalb besonders wichtig, da durch die Migration nicht a priori bestimmt werden kann, welche Fähigkeiten das Gerät haben wird, das die Ausführungskontrolle zum Zeitpunkt der Abarbeitung einer Aktivität besitzt. Ebenfalls durch die Migration ist begründet, dass nicht-funktionale Rahmenbedingungen in die Prozessbeschreibung aufgenommen werden müssen. Denn durch die so erlangte verteilte Ausführung der Prozesse ist

es möglich, dass die benutzerzentrierten Prozesse den Einflussbereich des Initiators verlassen, und nur durch eine adäquate enge Kopplung seiner Wünsche und Auflagen an den Prozess können diese Rahmenbedingungen auch während der Ausführung auf entfernten Systemen eingehalten werden.

Da die Ausführung und Migration von Mobilen Prozessen durch nichtfunktionale Parameter gesteuert wird und diese sich im Allgemeinen auf Aspekte des Ausführungskontextes beziehen, soll zudem ein Konzept eines generischen Kontextmodells sowie eines entsprechenden Verwaltungssystems entwickelt und umgesetzt werden. Hierbei müssen neben den etablierten Methoden zur Kontextnutzung und Adaption zusätzlich Verfahren und Konzepte zur Auflösung abstrakter Dienstklassen in entsprechende lokal oder entfernt ansprechbare Dienstinstanzen unter den nicht-funktionalen Rahmenbedingungen des Mobilen Prozesses erarbeitet und realisiert werden.

Die Umsetzung eines Konzepts zur kooperativen Ausführung von Prozessen in einer durch Heterogenität, Dynamik und variierende Kommunikationsbeziehungen geprägten mobilen Umgebung stellt mit ihren erweiterten Kontext- und Prozesskomponenten auch spezielle Anforderungen an eine grundlegende Kommunikationsinfrastruktur. Deshalb soll ein Konzept für eine Kommunikationskomponente erarbeitet und realisiert werden, die einen technologieunabhängigen und damit diskriminierungsfreien Transport der für die Ausführung Mobiler Prozesse notwendigen Daten ermöglicht. Hierzu soll ein Overlay-Netzwerk etabliert werden, dass beliebige darunter liegende Kommunikationsnetze und Transportprotokolle zu einem virtuellen Netzwerk koppelt und so einen Zugang der Mobilen Prozesse zu den Potenzialen der unterschiedlichen Systeme einer mobilen Umgebung ermöglicht.

1.3. Gang der Untersuchung

Wie in der Einleitung dargelegt, können bestehende Ansätze das Potenzial mobiler Umgebungen bisher nicht ausreichend dem Nutzer zugänglich machen. Vielmehr ist ihre Unterstützung zur Entwicklung und Ausführung von Softwaresystemen in der Regel auf eine ad-hoc-statische und anwendungsorientierte Ebene begrenzt. Eine Weiterentwicklung dieser Systeme im Sinn der Vision des Ubiquitous Computing verlangt jedoch, die Benutzer und ihre oftmals komplexen Aufgaben in den Fokus zu rücken und deshalb etablierte Middleware-Konzepte hin zu Plattformen zur Realisierung

flexibler und benutzerzentrischer Anwendungen weiterzuentwickeln. Hierzu wird in dieser Arbeit angeregt, komplexe Aufgaben der Nutzer abstrakt zu beschreiben und kooperativ durch die Geräte einer mobilen Umgebung ausführen zu lassen. Damit ist es das übergeordnete Ziel der vorliegenden Arbeit, eine kontextbasierte Kooperation durch die Migration und verteilte Ausführung von Prozessen im Umfeld mobiler Systeme zu entwickeln und – als Demonstration ihrer Umsetzbarkeit – auch prototypisch zu implementieren.

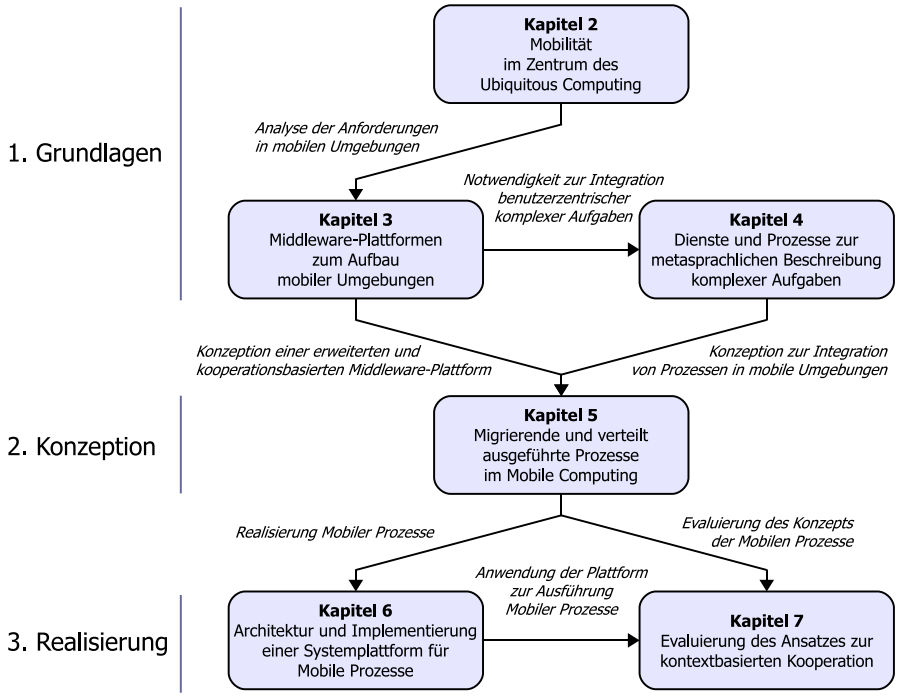


Abbildung 1.2.: Struktureller Aufbau der Arbeit

Damit gliedert sich die vorliegende Arbeit in die drei Hauptbereiche (a) der Darstellung und Analyse der *Grundlagen*, (b) der Entwicklung einer *Konzeption* und (c) die *Realisierung* der vorgeschlagenen Konzepte (vgl. Abbildung 1.2). Dabei werden zunächst die elementaren Entwurfskonzepte erarbeitet, um daraus im zweiten Schritt das konzeptionelle Rahmenmodell und dann dessen Überführung in ein Realisierungsmodell abzuleiten. Danach wird die prototypische Umsetzung des zuvor erarbeiteten Ansatzes dargestellt und evaluiert. Abbildung 1.2 verdeutlicht den entsprechenden strukturellen Aufbau der Arbeit und die Zusammenhänge zwischen den einzelnen Kapiteln noch einmal grafisch.

Dabei umfasst der erste Bereich sowohl konzeptionelle als auch technische Aspekte, die grundlegend für diese Arbeit sind. So führt Kapitel 2

zunächst in die elementaren Begriffe und Eigenschaften des Mobile Computings ein. Es werden hierbei die unterschiedlichen Facetten des Begriffs der Mobilität sowie die intrinsischen Eigenschaften mobiler Systeme dargelegt, und es wird eine Abgrenzung von traditionellen und mobilen Verteilten Systemen herausgearbeitet. Diese Analyse der Anforderungen und Eigenschaften der Mobilität bilden die Grundlage für Kapitel 3, in dem Middleware-Plattformen im Allgemeinen und diejenigen mit einem speziellen Fokus auf die Unterstützung mobiler Systeme untersucht werden. Hierbei werden zunächst die Anforderungen, die an solche Middleware-Systeme gestellt werden, herausgearbeitet. Darauf aufbauend werden bestehende Ansätze von kommunikationsorientierten und kontextbasierten Ansätzen eingeführt, exemplarische Vertreter vorgestellt und die sich aus der Untersuchung ergebenden Implikationen für die weitere Arbeit abgeleitet. Ein zentrales Ergebnis dieser Untersuchung ist, dass mit den bestehenden Systemen komplexe beziehungsweise strukturierte Aufgaben nicht ausreichend unterstützt werden. Deshalb führt das Kapitel 4 in – aus dem Umfeld dienstorientierter Architekturen entlehnte – Möglichkeiten ein, solche vielschichtigen und gegliederten Aufgaben in Form von abstrakten Diensten und Prozessen metasprachlich zu beschreiben.

Der zweite Bereich umfasst mit Kapitel 5 den konzeptionellen Teil der Arbeit. Hierbei wird systematisch auf der Basis der zuvor erarbeiteten Middleware-Klassen im Mobile Computing eine benutzerorientierte Erweiterung abgeleitet, welche die Heterogenität in mobilen Umgebungen nicht nur als technologische Herausforderung versteht, sondern auch als wertvolles Potenzial, das zur Ausführung komplexer Aufgaben genutzt werden kann. Auf der Basis dieses konzeptionellen Fundaments wird die *Kontextbasierte Kooperation* als übergeordnetes Modell zur gemeinschaftlichen Ausführung von komplexen Aufgaben in mobilen Umgebungen eingeführt. Aus diesem übergeordneten abstrakten Modell wird dann ein Realisierungskonzept abgeleitet, das durch die Migration von abstakt beschriebenen Prozessen eine verteilte und damit kooperative Ausführung von komplexen und benutzerzentrischen Aufgaben ermöglicht (*Mobile Prozesse*). Hierzu wird ein entsprechendes Prozessmetamodell konzeptionell erarbeitet und es wird aufgezeigt, wie der Kontext als ergänzende Wissensbasis für die Ausführung und Migration der Mobilen Prozesse dienen kann. Hierbei wird auch dargelegt, welche ergänzenden konzeptionellen Anforderungen ein solcher Ansatz an die Integration von Kontextwissen stellt und wie diese Anforderungen konzeptionell gelöst werden können. Zusätzlich wird herausgearbeitet, in welchem Umfang die aus traditionellen prozes-

sorientierten Systemen bekannten Konzepte zur Integration von transaktionalem Verhalten auf das Modell der Mobilen Prozesse übertragbar sind, wo die Grenzen einer solchen Überarbeitungsmöglichkeit liegen beziehungsweise welche konzeptionellen Ergänzungen dabei gemacht werden müssen.

Im dritten Bereich, der Realisierung, wird die im Projekt *Distributed Environment for Mobility-Aware Computing* (DEMAC) entwickelte prototypische Umsetzung der zuvor erarbeiteten Konzepte näher beschrieben. Hierzu werden in Kapitel 6 die in dem Projekt erarbeitete und implementierte Plattformarchitektur sowie ihre Komponenten vorgestellt. Dabei wird zunächst auf die durch ein *Overlay-Netzwerk* geprägte Kommunikationsplattform eingegangen, bevor das erweiterte Kontextsystem und die Komponente zur Integration Mobiler Prozesse eingeführt werden. Im Anschluss wird in Kapitel 7 untersucht, wie die Erwartungen an das Konzept der kontextbasierten Kooperation durch die Realisierung in Form von Mobilen Prozessen erfüllt werden. Hierzu wird das Konzept zunächst anhand der aufgestellten Anforderungen bewertet. Anschließend wird am Beispiel der prototypischen Implementierung des Projekts DEMAC die erwartete Steigerung der Ausführungswahrscheinlichkeit komplexer Aufgaben untersucht. Abschließend wird die Systemplattform anhand der während der Umsetzung implementierten Beispielanwendungen und -szenarios einer praktischen Evaluierung unterzogen.

Am Ende der Arbeit wird in Kapitel 8 eine Zusammenfassung und Bewertung der erzielten Ergebnisse vorgenommen, bevor ein Ausblick auf ergänzende und erweiterte Fragestellungen die Arbeit abschließt.

1.4. Ergebnisse und Beiträge

Im Rahmen der Untersuchungen und Analysen dieser Arbeit sowie der konzeptionellen Umsetzung der erarbeiteten Ergebnisse sind unterschiedliche Beiträge zur Wissenschaft und Praxis hervorgegangen, von denen an dieser Stelle die wichtigsten bereits vorweggenommen werden sollen. Hierbei kann als Gesamtergebnis festgehalten werden, dass es gelungen ist, die angestrebte konzeptionelle Erweiterung kontextbezogener Middleware-Plattformen hin zur Unterstützung echten kooperativen Verhaltens und deren technische Umsetzung mit Hilfe von Mobilen Prozessen zu realisieren. Die in diesem Zusammenhang erbrachten Beiträge betreffen dabei unter anderem folgende Teilergebnisse:

- *Konzeptionelles Modell der kontextbasierten Kooperation* (Abschnitt

5.3) – Ein Konzept zur Erweiterung kontextbezogener Middleware-Ansätze im Mobile Computing um kooperatives und benutzerzentrisches Verhalten zur Erschließung größerer Anteile des Potenzials einer mobilen Umgebung. [KZTL08]

- *Realisierungskonzept der Mobilen Prozesse* (Abschnitt 5.4) – Realisierungsmodell der kontextbasierten Kooperation in Form von migrierenden und verteilt ausgeführten Prozessen zur Ausführung komplexer Aufgaben im Mobile Computing auf der Basis eines dienstorientierten Ansatzes. [ZK07, KZL07b, KZL06]

- *Erweiterung des Prozesslebenszyklus* (Abschnitt 5.4.4) – Erweiterung des allgemeinen Prozesslebenszyklus zur Integration der Migration in eine Ausführungskomponente für Mobile Prozesse. [KZL07b]

- *Abstraktes Modell zur Beschreibung von Diensten* (Abschnitt 5.5.4) – Konzept zur technologieunabhängigen Integration heterogener Anwendungen und Dienste in Prozessbeschreibungen durch eine dreischichtige Dienstabstraktion bei gleichzeitiger Reduktion der Beschreibungskomplexität. [KZL07a]

- *Anwendungsunabhängiges und generisches Kontextmodell* (Abschnitt 5.6) – Konzeption eines generischen und anwendungsunabhängigen Kontextmodells und -managementsystems zur Unterstützung der Ausführung a priori unbekannter Aufgaben. [KZTL08]

- *Middleware-Architektur zur Realisierung der kontextbasierten Kooperation* (Kapitel 6) – Die DEMAC-Middleware implementiert eine Ausführungsumgebung für Mobile Prozesse auf Basis einer erweiterten kontextbasierten Systemplattform zur Etablierung eines technologieneutralen Overlay-Netzwerks für mobile Umgebungen. Die entwickelte Middleware-Plattform liefert dabei die praktische Systemumgebung zur Erprobung und Evaluierung des Konzepts der kontextbasierten Kooperation und seiner Umsetzung als Mobile Prozesse. [KZL07b, KZL06, Kun05a]

2. Mobilität im Zentrum des Ubiquitous Computing

Durch die fortschreitenden Innovationen in der Informations- und Kommunikationstechnologie haben sich PCs heutzutage weit verbreitet (vgl. Abschnitt 1). Jedoch stellt sich die Frage, wie der Computer der Zukunft aussehen wird. Bereits 1991 hat *Mark Weiser*[1] die Vision entwickelt, dass auf das Zeitalter der Personal Computer die Ära der unsichtbaren, kooperierenden und überall verfügbaren Rechner in Form des *Ubiquitous Computings* folgen wird. [Mat01, Wei91] Dabei soll die Computertechnologie nicht mehr bewusst genutzt werden, sondern in den Hintergrund treten und eher Mittel zum Zweck werden:

> ***Ubiquitous computing*** *has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.*
>
> [Wei93, Seite 71]

Neben dem Begriff des Ubiquitous Computings hat sich zunehmend der häufig äquivalent gebrauchte Begriff des *Pervasive Computings* etabliert. Dieser mehr von der Industrie geprägte Begriff steht, im Gegensatz zu der wissenschaftlichen Idee der humanzentrischen und unaufdringlichen Unterstützung durch Computertechnologie von Weiser, eher für eine überall verfügbare Technik, die elektronischen Handel und webbasierte Geschäftsprozesse fördern. [Mat01]

Dass jedoch die Vision des Ubiquitous Computings immer mehr zur Realität wird, zeigt sich nicht nur durch die stetige Miniaturisierung von Prozessoren und Speicher, sondern auch durch die zunehmende Vernetzung und Integration von Computertechnologie in Alltagsgegenstände (*Embedded Systems*), die so allmählich zu immer intelligenteren Geräten (*Smart Devices*) werden. Aber auch Objekte ohne enthaltene Prozessoren können, zum Beispiel durch *RFID-Etiketten*, eine elektronische Identität bekommen, um sie mit beliebigen Daten zu verknüpfen. So können *virtuelle Datenschatten* aller physischen Objekte entstehen, die Auskunft über die Historie oder Zusammensetzung geben oder die Schlüssel zu erweiterten Dienstleistungen darstellen. [Mat05b, Mat03]

[1] * 23. Juli 1952, † 27. April 1999

Ein weiterer Schritt hin zum *„globalen ubiquitären Computer“* [Mil06, Seite 385] ist die Konvergenz der Kommunikationsnetzwerke hin zu globalen digitalen Netzwerken, in denen Daten-, Sprach-, und Multimediadienste zusammengeführt werden. So kann zum Beispiel über einen breitbandigen Internetzugang bereits zusätzlich auch telefoniert (*VoIP*) und ferngesehen (*Videostreaming*) werden. Zudem werden nicht nur Geräte an diesem allgegenwärtigen Computer teilnehmen, sondern auch Sensoren, die die Eigenschaften der Umgebung wahrnehmen und bereitstellen, sowie die Nutzer mit ihren Bedürfnissen. Da die Wünsche und Anforderungen der Nutzer als nicht-funktionaler Teil des Systems angesehen werden, müssen Entscheidungen, die der Nutzer treffen würde, nun vom System selbst getroffen werden. [Mil06, Mat05b, VC99]

Ein immanenter Bestandteil eines solchen ubiquitären Systems ist die Mobilität: Mobile Benutzer benötigen dabei zu jeder Zeit und von jedem Ort aus Zugriff auf ihre Anwendungen und Daten. Zudem sind viele Geräte mobil und somit fähig, dynamische Beziehungen einzugehen, um ihre eigenen Fähigkeiten durch Kooperation mit der Umgebung zu erweitern. Aber auch Software kann mobil sein, indem sie auf andere Geräte der mobilen Umgebung migriert und dann dort ausgeführt wird. [Mil06, Kun05a]

Mit den Eigenschaften und technologischen Gegebenheiten solcher mobilen Systeme befasst sich das folgend beschriebene Teilgebiet des *Mobile Computings*. Dabei wird der Begriff der Mobilität näher untersucht, Gemeinsamkeiten und Unterschiede von traditionellen und mobilen Verteilten Systemen dargelegt sowie auf die intrinsischen Eigenschaften mobiler Systeme eingegangen. Zudem werden technologische und architektonische Charakteristika der Kommunikation mobiler Systeme beschrieben. Abschließend werden die Anforderungen von Anwendungen in einem mobilen Umfeld dargestellt.

2.1. Mobile Computing

Verteilte Systeme sind, gerade im Umfeld des Ubiquitous Computings, zunehmend mit Aspekten der Mobilität konfrontiert (vgl. Abschnitt 2). Deshalb hat sich das Forschungsgebiet des *Mobile Computings* herausgebildet, das sich mit den Einflüssen der Mobilität auf Anwendungen, unterstützende Middleware-Plattformen, Betriebssysteme und die Hardware mobiler Systeme befasst. Dabei wird die Definition des Mobile Computing oftmals recht eng an den Begriff der Portabilität – also der leichten Beweglichkeit in der physikalischen Welt – angelehnt:

> ***Mobile computing systems*** *are computing systems that may be easily moved physically and whose computing capabilities may be used while they are being moved.*
>
> [BFar05, Seite 3]

Dabei werden jedoch wichtige Gesichtspunkte, wie die Einflüsse und Anforderungen mobiler Benutzer oder mobilen Codes nicht berücksichtigt. Deshalb ist eine allgemeinere Definition sinnvoll, die die Herausforderungen der Mobilität von beliebigen Entitäten eines Computersystems herausstellt:

> ***Mobile Computing*** *[is defined by a] [...] system in which computational components may change location.*
>
> [RPM00, Seite 245]

Damit stellt das Mobile Computing ein allgemeines Forschungsgebiet über die Mobilität in Computersystemen dar, unter dem sich je nach Betrachtungsschwerpunkt weitere Begriffe herausgebildet haben. So haben sich Teilgebiete mit eher technischem Fokus, wie unter anderem das *Embedded Computing*, *(Mobile) Ad-hoc-Netzwerke* (vgl. Abschnitt 2.3.2.1) oder *drahtlose Sensornetzwerke*, herausgebildet. Andere Gebiete, wie beispielsweise das *Nomadic Computing* oder die *Ambient Intelligence*, stellen dagegen eher einzelne Akteure oder Situationen in den Mittelpunkt.

Dabei befasst sich das Embedded Computing mit der Integration von meist kleinen und preiswerten Prozessoren, Speichermodulen, Sensoren und Kommunikationskomponenten in Alltagsgegenstände. Damit ist es diesen dann „intelligenten“ Geräten möglich, ihre Umgebung wahrzunehmen, die so erhaltenen Informationen sinnvoll zu verarbeiten und mit anderen Geräten in der Umgebung zu kooperieren. Drahtlose Sensornetzwerke hingegen versuchen, eine Brücke zwischen der realen und der virtuellen Welt zu schlagen, indem sie Umgebungsinformationen überwachen und diese Informationen anderen zur Verfügung stellen. Sensoren und Sensornetze stellen somit die „*Sinnesorgane*“ [Mat06, Seite 9] eingebetteter und mobiler Systeme dar. Dabei ist eine Kooperation der Systeme lediglich dann möglich, wenn die einzelnen Systeme miteinander kommunizieren können und auch in Situationen, in denen keine feste Kommunikationsinfrastruktur und keine zentrale Administration vorhanden sind. Mit den Herausforderungen, die sich unter solchen Bedingungen stellen, befasst sich das Teilgebiet der (Mobilen) Ad-hoc-Netzwerke. [Mat06, RKM02]

Auf diesen technologischen Grundlagen können dann Konzepte und Systeme entstehen, die spezielle Einsatzbereiche abdecken. Dazu gehören Sy-

steme, die die Bedürfnisse mobiler Benutzer (*Nomaden*) in einer transparenten, integrierten und komfortablen Weise unterstützen. Solche Architekturen und Plattformen werden deshalb auch unter dem Begriff des Nomadic Computings zusammengefasst. Andere Konzepte fokussieren sich auf die Unterstützung der Menschen in ihrem Alltag, indem eingebettete Systeme und deren Dienste mit Hilfe von einfachen und intuitiven Bedienoberflächen gesteuert werden können. [CCL03, Kle96]

Diese und andere Teilgebiete im Mobile Computing stehen im Allgemeinen eng miteinander in Beziehung und haben deshalb meist ähnliche oder gleiche Grundlagen, Anforderungen und Einschränkungen. Diese zu kennen und bei einem Systementwurf zu berücksichtigen ist, deshalb besonders wichtig.

2.2. Mobilität in der Informationstechnologie

Die Betrachtung der *Mobilität im Mobile Computing* bezieht sich im Allgemeinen darauf, welche Einheiten in einem System ihre Lokation wechseln können (vgl. Abschnitt 2.1). Dabei werden die Untergliederungen je nach Blickwinkel und Fokus anhand unterschiedlicher Gesichtspunkte vorgenommen.

Eine der grundlegenden Unterscheidungen ist dabei die Klassifizierung nach der Stofflichkeit der mobilen Einheiten in die Klassen der logischen und der physikalischen Mobilität (vgl. Abschnitt 2.2.1). Andere Mobilitätsklassen stellen hingegen einzelne Entitäten beziehungsweise deren Beziehung zu anderen in den Vordergrund (vgl. Abschnitt 2.2.2).

2.2.1. Physikalische und logische Mobilität

Die Einteilung von Mobilität in die Klassen der *physikalischen Mobilität* und der *logischen Mobilität* basiert auf der Unterscheidung zwischen materiellen und immateriellen Komponenten eines mobilen Systems. Dabei ist die physikalische Mobilität durch die Bewegung in der körperlichen Welt begründet (vgl. Abbildung 2.1(a)). Die logische Mobilität hingegen bezeichnet die Migration virtueller Einheiten (von Code) zwischen zwei Hosts (vgl. Abbildung 2.1(b)). [RPM00]

Beiden Mobilitätsausprägungen ist gemein, dass sich mit der Änderung des Aufenthaltsortes das Ausführungsumfeld (*Context*) verändert. Dies äußert sich vor allem darin, dass sich die verfügbaren Dienste und Ressourcen ändern. Aber auch nicht-funktionale Aspekte wie die Art des Zugriffs auf Dienste und Komponenten (*lokal* beziehungsweise *entfernt*) oder

(a) Physikalische Mobilität (b) Logische Mobilität

Abbildung 2.1.: Fundamentale Mobilitätsklassen

die Dienstgüte (*Quality of Service*) können sich im Zuge des Ortswechsels ändern.

Ein besonderer Unterschied besteht zwischen logischer und physikalischer Mobilität aus Sicht des *Software Engineerings*: Physikalische Mobilität stellt hier eine *Anforderung* an ein System dar, wogegen die logische Mobilität als *Entwurfsalternative* für ein System angesehen wird. [RPM00]

Diese Klassifizierung der Mobilität stellt jedoch keine disjunkte Zerlegung der Menge der mobilen Systeme dar. Im Gegenteil, die Vereinigung von logischer und physikalischer Mobilität in einem System kann durchaus sinnvoll sein. So kann zum Beispiel ein mobiler Softwareagent auf einem portablen PDA ausgeführt werden, sodass das Gesamtsystem beide Mobilitätsarten verbindet.

2.2.2. Klassifizierung bezüglich mobiler Einheiten

Betrachtet man die Mobilität bezüglich der durch sie beeinflussten Entitäten, kann eine diversifiziertere Klassifizierung als die in logische und physikalische Mobilität (vgl. Abschnitt 2.2.1) vorgenommen werden. In der Literatur haben sich in diesem Zusammenhang die Klassen der *Endgeräte-, Benutzer-*, *Dienst-*, *Session-* und *Netzwerkmobilität* herausgebildet. Auch bei dieser Einteilung handelt es sich nicht um disjunkte Klassen, da mobile Systeme, aus unterschiedlichen Perspektiven betrachtet, durchaus zu mehreren Klassen gehören können.

2.2.2.1. Endgerätemobilität

Endgerätemobilität liegt dann vor, wenn ein Gerät auch dann mit anderen vernetzt bleibt und somit kommunizieren kann, wenn es seinen Aufenthaltsort wechselt. So liegt zum Beispiel bei Mobiltelefonen Endgerätemobilität vor, da diese auch bei Bewegung innerhalb und zwischen Funkzellen ihre Dienstleistung aufrechterhalten. [Rot02a, SS02]

Im Zusammenhang mit der Edngerätemobilität wird oft auch von *Portabilität* gesprochen. Diese beiden Begriffe sind jedoch nicht synonym, da Geräte zwar portabel – also leicht tragbar – sein können, ohne jedoch dabei kommunizieren können zu müssen. Es handelt sich also vielmehr bei der Endgerätemobilität um eine besondere Ausprägung der Portabilität.

2.2.2.2. Benutzermobilität

Bei der *Benutzermobilität* oder auch *Personenmobilität* ist es einem Anwender möglich, eigene Kommunikationsdienste von beliebigen Geräten und Netzwerken aus zu nutzen. Dabei ist vor allem die eindeutige Identifizierung des Benutzers wichtig, um die Dienste dem legitimen Benutzer zuzuordnen. [Rot02a, SS02]

In Krankenhäusern werden beispielsweise Chipkarten zur Authentifizierung gegenüber der Telefonanlage eingesetzt. Dies ermöglicht es, Patienten auch dann stets unter derselben Rufnummer erreichbar zu sein, wenn sie in ein anderes Zimmer verlegt werden und die Gespräche somit an verschiedene Telefone geleitet werden müssen.

2.2.2.3. Dienstmobilität

Können auf personalisierte Dienste von beliebigen Orten immer in gleicher Art und Weise zugegriffen werden, so bezeichnet man dies als *Dienstmobilität*. Dies ist zum Beispiel dann der Fall, wenn ein Benutzer seine E-Mail von jedem beliebigen Ort aus über eine Web-Schnittstelle abrufen kann. [Rot02a, SS02]

2.2.2.4. Sessionmobilität

Von *Sitzungsmobilität* beziehungsweise *Sessionmobilität* spricht man dann, wenn von verschiedenen Geräten mit unterschiedlichen Eigenschaften auf den Zustand einer aktiven Sitzung zugegriffen werden kann, ohne dass die Mobilität der Geräte, der Benutzer oder der Anwendung zu unerwünschten Unterbrechungen beziehungsweise Abbrüchen führt. Hierbei muss vor allem die *Konsistenz der Sitzungsdaten* gewährleistet werden. [FHLM01]

Wird zum Beispiel eine Lieferroute am PC eines Disponenten erarbeitet und diese Daten dann zur Navigation im LKW verwendet, so spricht man hier von Sessionmobilität.

2.2.2.5. Netzwerkmobilität

Netzwerkmobilität bezeichnet die Fähigkeit eines drahtlosen Netzwerkes, sich selbst in der physikalischen Umgebung zu bewegen. Dies ist vor allem dann der Fall, wenn das Netzwerk ohne feste Infrastrukturkomponenten als Ad-hoc-Netzwerk aufgebaut ist. [SS02]

Beispielhaft hierfür sind Peer-to-Peer-Netzwerke zwischen Fahrzeugen, in denen Gefahrenmeldungen, wie das Erkennen von Aquaplaning, und andere fahrtspezifischen Informationen an nachfolgende Fahrzeuge weitergegeben werden können. Diese Netzwerke bestehen zwar nur temporär, bewegen sich aber mit den beteiligten Fahrzeugen und besitzen somit die Eigenschaft der Netzwerkmobilität.

2.3. Abgrenzung traditioneller und mobiler *Verteilter Systeme*

Die Vernetzung von Rechensystemen ist heutzutage längst nicht mehr die Ausnahme, vielmehr ist sie zur Regel geworden. Zu diesem Trend hat vor allem der Einsatz von *Verteilten Systemen* beigetragen, welche die Vernetzung von Rechensystemen für Anwendungen und Anwender transparent gestalten. Deshalb definieren *Tanenbaum* und *van Steen* den Begriff des Verteilten Systems auch genau über diese Eigenschaft:

> *A **distributed system** is a collection of independent computers that appears to its users as a single coherent system.*
>
> [TS02, Seite 2]

Diese Definition beinhaltet dabei zwei Hauptmerkmale, die ein Verteiltes System auszeichnen. Dies ist zum einen der Einsatz von autonomen und heterogenen Rechnern und zum anderen die Verwendung von Softwarestrukturen, die diese einzelnen Komponenten des Netzwerks als ein einzelnes System erscheinen lassen. [TS02]

Eine der Hauptmotivationen für den Einsatz von Verteilten Systemen ist die *gemeinsame Nutzung von Ressourcen*. Zudem bietet solch eine kooperative Architektur im Allgemeinen eine *höhere Verfügbarkeit*, da der Ausfall einzelner Systemkomponenten oftmals kompensiert werden kann oder nur einzelne Teile des Systems betrifft. Ein weiterer Vorteil ist die meist vorhandene *Offenheit* Verteilter Systeme, die es erlaubt, Komponenten leicht hinzuzufügen oder zu ersetzen. Außerdem besitzen solche Systeme meist eine gute *Skalierbarkeit*, also die Fähigkeit mit steigender Anzahl von Benutzern und Komponenten immer noch effizient zu funktionieren. [TS02, CDK01]

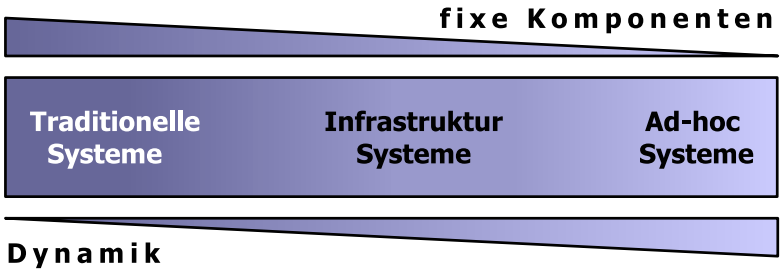


Abbildung 2.2.: Arten Verteilter Systeme

In zunehmenden Maß werden immer mehr mobile Geräte in diese traditionellen Verteilten Systeme integriert (*Infrastruktur-Systeme*) beziehungsweise bilden eigenständige, spontan zusammengefundene *mobile Ad-hoc-Netzwerke* (vgl. Abbildung 2.2). Diese beiden Begriffe werden auch unter der Bezeichnung *Mobile Computing* zusammengefasst (vgl. Abschnitt 2.1) und bilden den infrastrukturellen Rahmen, in dem sich *Mobile Verteilte Systeme* bewegen. Dabei kann meist festgestellt werden, dass die Dynamik der Systeme umso größer wird, je weniger sie an fixe Infrastrukturkomponenten gebunden sind.

2.3.1. Eigenschaften traditioneller Systemarchitekturen

Eine der grundlegenden Eigenschaften von Verteilten Systemen ist die *Fähigkeit zur Kommunikation* zwischen den an ihnen beteiligten autonomen Computersystemen. Diese trivial erscheinende Eigenschaft bedingt aber viele weitere Eigenschaften, die Verteilte Systeme auszeichnen. Dies sind zum Beispiel der einfache *Zugriff auf entfernte Ressourcen* oder die Unterstützung der *Zusammenarbeit* und des *Informationsaustauschs* zwischen den Systemkomponenten. [TS02, Lam94]

Die Autonomie der beteiligten Rechensysteme führt dazu, dass Verteilte Systeme im Allgemeinen einen hohen Grad an *Heterogenität* aufweisen. Dabei müssen unterschiedliche Netzwerke, Computer-Hardware, Betriebssysteme, Programmiersprachen und Dienstimplementierungen miteinander zusammenarbeiten. Dies bedingt, dass Verteilte Systeme ihre Ressourcen und Dienste über normierte Schnittstellen und Protokolle untereinander verfügbar machen. Diese *Neutralität der Schnittstellen* führt dann zur *Offenheit* Verteilter Systeme, also der Fähigkeit, Ressourcen problemlos zu ersetzen beziehungsweise hinzuzufügen. Die Offenheit ist zudem eine Voraussetzung für die im Allgemeinen gute *Skalierbarkeit* in Bezug auf die Größe des Systems, der geografischen Verteilung der Systemkomponenten

und der Administrierbarkeit. [TS02, CDK01]

Eine der wichtigsten Eigenschaften Verteilter Systeme ist, die Auswirkungen der Verteilung für Anwendungen und Anwender zu verbergen. Diese Eigenschaft der *(Verteilungs-)Transparenz* gliedert sich dabei in verschiedene Aspekte (siehe Tabelle 2.1), die zusammen den Grad der Transparenz des Gesamtsystems bestimmen und damit einen hohen Einfluss auf den Entwurf der enthaltenen Komponenten besitzen. [TS02, CDK01]

Transparenzaspekt	Beschreibung
Zugriffstransparenz	Ermöglicht die einheitliche Darstellung und den gleichförmigen Zugriff auf Daten im Verteilten System.
Ortstransparenz	Verbirgt die physikalische Position einer Ressource im System.
Migrationstransparenz	Ermöglicht einen einheitlichen Zugriff auf eine Ressource, auch wenn sich ihre physikalische Position ändert.
Relokationstransparenz	Ermöglicht die Veränderung einer Ressource auch während ihrer Nutzung.
Replikationstransparenz	Verbirgt das mehrfache Vorhalten von Kopien einer Ressource.
Nebenläufigkeitstransparenz	Verbirgt die Auswirkungen, die konkurrierende Zugriffe mehrerer Benutzer auf eine Ressource erzeugen.
Fehlertransparenz	Verbirgt den Ausfall und die Wiederherstellung von Ressourcen.
Persistenztransparenz	Maskiert, ob eine Ressource sich zurzeit im flüchtigen oder persistenten Zustand befindet.

Tabelle 2.1.: Aspekte der Verteilungstransparenz (nach (TS02))

Insgesamt bieten Verteilte Systeme durch die meist gegebene inhärente Redundanz von Systemkomponenten sowie ihrer Offenheit einen *Gewinn an Zuverlässigkeit und Verfügbarkeit* des Systems. Zudem lässt sich durch die gemeinsame Nutzung von Ressourcen oftmals ein *Kostenvorteil* erzielen (zum Beispiel durch die gemeinsame Nutzung eines Druckers). Jedoch können auch – gerade bei fehlender Redundanz – Abhängigkeiten zwischen den Systemkomponenten entstehen, die in Fehlerfällen die entstehenden Effekte im System weiter propagieren können. Zudem entstehen durch die massive parallele Arbeitsweise der Systeme weitere Herausforderungen

bezüglich Synchronisation der Dienstzugriffe und Konsistenz der Daten. [TS02, Lam94]

2.3.2. Eigenschaften mobiler Systemarchitekturen

Mobile Systeme besitzen, da sie ebenfalls Verteilte Systeme sind, die meisten Eigenschaften, die im Abschnitt 2.3.1 beschrieben sind. Auch ihre primäre Eigenschaft ist es, die *Kommunikation* zwischen den mobilen Komponenten zum *Zugriff auf entfernte Ressourcen* und zum *Informationsaustausch* zu ermöglichen. Da gerade im mobilen Umfeld die *Heterogenität* der Systemkomponenten groß ist, müssen mobile Systemarchitekturen ebenfalls ein großes Maß an *Offenheit* bieten. Zudem wird gleichermaßen ein gewisser Grad an *Verteilungstransparenz* angestrebt, auch wenn teilweise ganz bewusst auf einzelne Aspekte verzichtet wird (vgl. Abschnitt 2.5.2).

Zunächst werden in Tabelle 2.2 und den folgenden Abschnitten infrastrukturbasierte und Ad-hoc-Systeme vorgestellt und verglichen. Danach wird auf die speziellen und intrinsischen Eigenschaften mobiler Systeme eingegangen, die zusätzliche Rahmenbedingungen im Mobile Computing darstellen.

Infrastruktur-Systeme	Ad-hoc-Systeme
Bestehen aus festen, örtlich abgegrenzten Zellen und Basisstationen.	Kommen ohne feste Basisstationen aus und sind sehr kurzfristig einsatzfähig.
Beruhen auf einem statischen Backbone-Netz.	Besitzen eine sehr dynamische Netzwerktopologie mit Multi-Hop-Wegewahl.
Besitzen eine relativ zuverlässige Umgebung mit stabilen Verbindungen.	Besitzen eine unzuverlässige Umgebung mit nur sporadischen Verbindungen.
Benötigen eine detaillierte Planung vor dem Aufbau der Infrastruktur.	Formieren sich automatisch und passen sich Veränderungen an.

Tabelle 2.2.: Vergleich von Infrastruktur- und Ad-hoc-Systemen (nach (RR02))

2.3.2.1. Ad-hoc-Systeme

Ad-hoc-Systeme bestehen allgemein aus mobilen Knoten, die sich dynamisch und autonom zu drahtlosen Kommunikationsnetzwerken zusammenschließen und dabei auf zentrale Infrastrukturkomponenten und Ad-

ministration verzichten. [CCL03] Allgemein können Ad-hoc-Netzwerke wie folgt definiert werden:

> *An **ad hoc network** is a (possibly mobile) collection of communication devices (nodes) that wish to communicate, but have no fixed infrastructure available, and have no predetermined organization of available links.*
>
> [RR02, Seite 20]

Knoten in Ad-hoc-Systemen sind also zunächst autonom und müssen eigenständig ermitteln, welche anderen Knoten in ihrer Kommunikationsreichweite sind. Nach erfolgreichem Kommunikationsaufbau sind sie dann in der Lage, mit den ihnen unmittelbar verbundenen Geräten Daten auszutauschen (vgl. Abbildung 2.3). Damit auch Knoten miteinander kommunizieren können, die nicht direkt miteinander verbunden sind, werden Zwischenknoten benötigt, die im Auftrag des Senders dessen Datenpakete weiterleiten beziehungsweise ausliefern (*Routing*). Da die Wegewahl in solchen Netzwerken somit über Zwischenstopps führt, werden sie auch als *Multi-Hop-Netzwerke* bezeichnet. [CCL03, RR02]

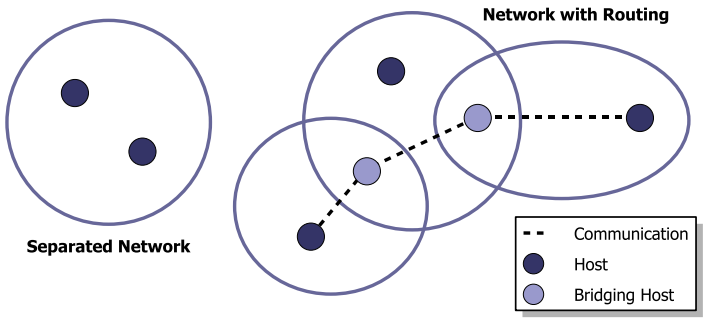


Abbildung 2.3.: Strukturübersicht von Ad-hoc-Netzwerken

Da jeder Knoten autonom ist und sich frei bewegen kann, haben Ad-hoc-Systeme meist eine sehr dynamische und unvorhersagbare Netzwerktopologie. Zudem können sich die Konnektivität des Gerätes und die Qualitätseigenschaften aktiver Verbindungen ständig ändern, sodass solche Netze eine geringere Zuverlässigkeit für Anwendungen und Nutzer bieten. Des Weiteren ist der Betrieb eines Ad-hoc-Systems meist mit einem – verglichen mit anderen drahtlosen Kommunikationsformen – höheren Energieverbrauch verbunden, da die einzelnen Endgeräte alle Aufgaben zur Steuerung des Medienzugriffs vorhalten müssen. Außerdem benötigt die Weiterleitung fremder Datenpakete zusätzliche Energie, die dann eigenen Anwendungen nicht mehr zur Verfügung steht. [Sch03, CCL03, RR02]

2.3.2.2. Infrastruktur-Systeme

Infrastruktur-Systeme nehmen als hybride Systemform eine Brückenstellung zwischen traditionellen und Ad-hoc-Systemen ein (siehe Abbildung 2.2). Solche Infrastruktur-Systeme werden meist durch zellulare Netzwerke realisiert. Dabei wird jede *(Funk-)Zelle* durch einen Zugangspunkt (*Access Point*) – auch Basisstation genannt – repräsentiert. Dabei dient der Access Point nicht nur zur Steuerung des Medienzugriffs, sondern auch als Übergang in andere drahtgebundene oder -lose Netzwerke (siehe Abbildung 2.4). Werden mehrere Access Points verbunden, können diese ein größeres logisches Netz bilden. Beispiele für Infrastruktur-Systeme sind aktuelle GSM- und UMTS-Mobilfunknetze sowie WLAN-Netze (vgl. Abschnitt 2.5.3.2). [IM05, Sch03]

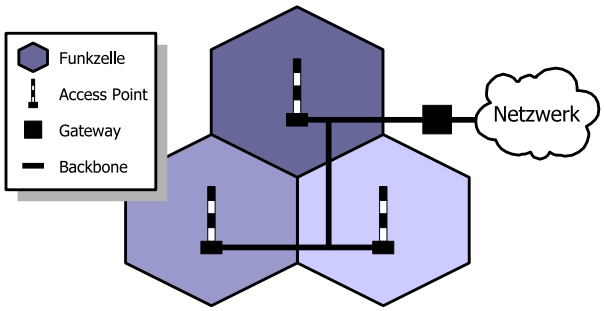


Abbildung 2.4.: Aufbau eines zellularen Netzwerks

Zum Aufbau von Infrastruktur-Systemen ist im Gegensatz zu Ad-hoc-Systemen eine detaillierte Planung notwendig. Jedoch kann zugleich ein großer Teil der notwendigen und relativ komplexen Steuerung des Netzwerkzugriffs in den Zugangspunkt verlegt werden, sodass die mobilen Endgeräte relativ einfach gehalten werden können. Zudem ist es durch die Steuereigenschaft des Access Points ebenfalls leichter möglich, Dienstgütegarantien abzugeben. Jedoch verliert man auch ein Stück an Flexibilität, da man auf die Infrastruktur angewiesen ist und deshalb bei deren Ausfall keine mobile Kommunikation mehr möglich ist. [Sch03, RR02]

2.4. Intrinsische Eigenschaften mobiler Systeme

Mobile Systeme sind zwar ebenfalls Verteilte Systeme, haben aber zusätzlich eigene Eigenschaften und Einschränkungen, denen sie unterliegen. Diese zu kennen sowie beim Entwurf und Umgang mit mobilen Systemen zu berücksichtigen, ist essenziell, da sie einen intrinsischen Rahmen im Mobile Computing bilden.

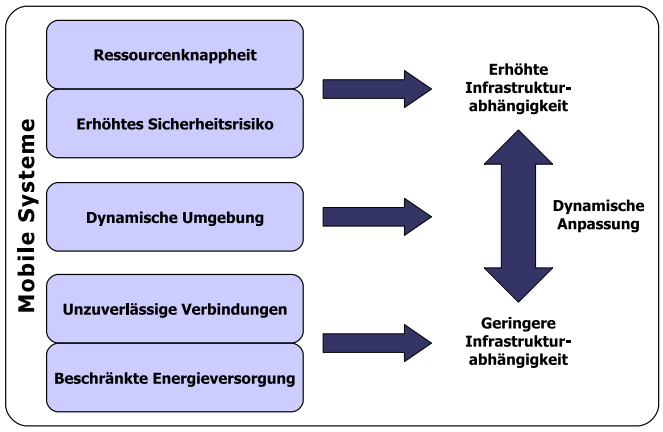


Abbildung 2.5.: Eigenschaften mobiler Systeme (nach (AGRS05))

Die *Ressourcenknappheit* mobiler Systeme und ihr *erhöhtes Sicherheitsrisiko* führen eher zu Systementwürfen mit stärkerer Infrastrukturabhängigkeit. Dem gegenüber stehen jedoch *unzuverlässige Verbindungen* und *beschränkte Energiequellen*, die wiederum eher eine Unabhängigkeit von statischen Infrastrukturkomponenten fordern (siehe Abbildung 2.5). Dieses Spannungsfeld zusammen mit der sich dynamisch ändernden Umgebung mobiler Systeme ist Ausgangspunkt des Wunschs nach Plattformen und Anwendungen, die sich dynamisch den jeweiligen Anforderungen und Bedingungen im System anpassen. Dazu muss jedoch die in traditionellen Verteilten Systemen stark ausgeprägte Verteilungstransparenz zu Gunsten von *Awareness* und *Adaptability* eingeschränkt werden (vgl. Abschnitt 2.5.2). [AGRS05]

2.4.1. Ressourcenknappheit

Auch wenn sich mobile Systeme stetig weiterentwickeln und immer leistungsfähiger werden, so sind diese doch stets im Verhältnis zu stationären Rechensystemen leistungsärmer. Mobile Systeme müssen, da sie von endlichen Energiequellen (beispielsweise Batterien) abhängen, so entworfen werden, dass sie möglichst effizient und sparsam mit der zur Verfügung stehenden Energie umgehen. Zudem muss bei mobilen Systemen die Größe der verwendeten Komponenten berücksichtigt werden, was dazu führt, dass nur hoch integrierte Elemente mit ihren besonderen physikalischen Grenzen Einsatz finden können. Daraus resultiert im Allgemeinen, dass die CPU und der Speicher mobiler Geräte limitiert sind. [BFar05, AGRS05, Sat96]

2.4.2. Erhöhtes Sicherheitsrisiko

Die Portabilität und die deshalb gebotene geringere Größe von mobilen Geräten birgt in sich ein erhöhtes Risiko, dass die Geräte verloren oder gestohlen werden. Aber auch das Risiko, dass Unberechtigte Zugriff auf die Geräte mit ihren Daten und Anwendungen erlangen, ist wesentlich höher als bei stationären Geräten. [AGRS05, Sat96]

2.4.3. Variierende Konnektivitätsbedingungen

Mobile Systeme stützen sich im Allgemeinen auf drahtlose und damit unzuverlässige Kommunikationsverbindungen mit sich ständig ändernden Qualitätsparametern (QoS). Dies basiert darauf, dass die Performanz der Verbindungen in Bezug auf Bandbreite und Latenz stark variiert und zusätzlich Verbindungsabbrüche, ob gewollt oder ungewollt, verbreitet auftreten (vgl. Abschnitt 2.5.3). [BFar05, AGRS05, Sat96]

2.4.4. Abhängigkeit von endlichen Energiequellen

Eine weitere Eigenschaft von nahezu allen mobilen Systemen ist, dass sie auf endliche Energiequellen (meist in Form von Batterien oder Akkus) angewiesen sind. Dieser Umstand hat Einfluss auf nahezu alle Komponenten eines mobilen Gerätes, da immer eine Balance zwischen Energieverbrauch beziehungsweise -speicherkapazität und den gewünschten Anforderungen gefunden werden muss. Dies betrifft aber nicht nur die Hardware eines mobilen Gerätes, sondern auch Software-Einheiten, die durch effiziente Ausführung ihrer Aufgaben die vorhandenen Energiereserven schonen können. [BFar05, Sat96]

2.4.5. Heterogenität der Systemlandschaften

Durch die eingeschränkten Ressourcen haben sich viele spezialisierte Geräte im Umfeld des Mobile Computings entwickelt. Diese reichen vom Mobiltelefon oder PDA bis hin zu Sensornetzwerken und so genanntem *Smart Dust*. Dabei haben sich durch kurze Produktlebenszyklen und die unterschiedlichen Einsatzschwerpunkte eine Vielzahl von verschiedenen Systemen und Systemplattformen gebildet, sodass die Umgebung eines mobilen Gerätes durch eine große Heterogenität der enthaltenen Systeme gekennzeichnet ist. [BFar05]

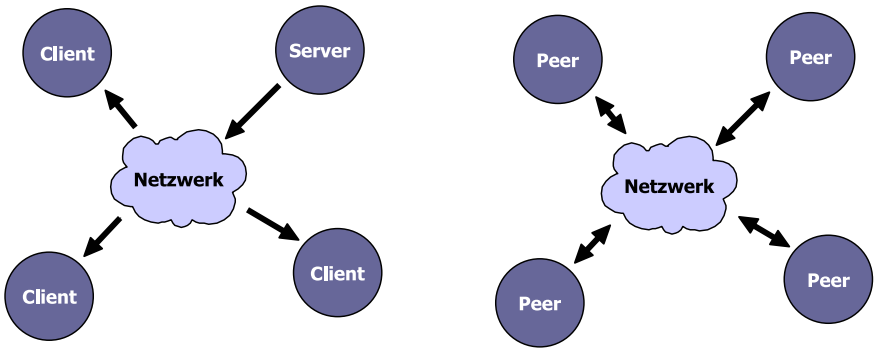
2.5. Kommunikation im Mobile Computing

Eine der wichtigsten Eigenschaften vieler mobiler Systeme ist deren Fähigkeit zur Kommunikation. Dabei werden mobile Geräte im Allgemeinen drahtlos direkt mit anderen Geräten oder mit Infrastrukturkomponenten verbunden. [Rot02a] Deshalb werden in den folgenden Abschnitten Architekturmodelle, besondere Anforderungen und Technologien für die Kommunikation im Mobile Computing vorgestellt.

2.5.1. Spannungsfeld zwischen asymmetrischen und symmetrischen Architekturmodellen

Kommunikationsarchitekturmodelle beschreiben für ein Verteiltes System, welche Rollen die enthaltenen Komponenten einnehmen und in welcher Beziehung sie zueinander stehen. [Ham05] Diese sind, da sie die Kommunikation auf Anwendungsebene beschreiben, unabhängig von den darunter liegenden Netzwerktechnologien und Infrastruktur-Systemen (vgl. Abschnitte 2.5.3 und 2.3.2.2).

Die heutzutage dominierenden Architekturmodelle sind die in den folgenden Abschnitten beschriebenen Client-Server- und Peer-to-Peer-Modelle (vgl. Abbildung 2.6). Daneben gibt es natürlich auch hybride Formen, die die Aspekte beider Modelle in unterschiedlichen Ausprägungen vereinen.



(a) Client-Server-Modell (b) Peer-to-Peer-Modell

Abbildung 2.6.: Kommunikationsmodelle in Verteilten (mobilen) Systemen

Gerade bei mobilen Systemen ist die Wahl eines Architekturmodells meist nicht nur durch die zu erbringende Aufgabe begründet, sondern muss auch die Eigenschaften des Einsatzumfeldes der Anwendung berücksichtigen. Dabei ist der Grad der Dynamik des Applikationskontextes ein wichtiger Faktor, da dieser angeben kann, ob die Zentralisie-

rung von Diensten sinnvoll ist. So ist zum Beispiel eine reine Client-Server-Architektur in einem hoch-dynamischen Ad-hoc-Netzwerk unter unzuverlässigen Kommunikationsbedingungen im Allgemeinen weniger leistungsfähig, als eine Peer-to-Peer-Variante, die alle benötigten Dienste mehrfach repliziert vorhält.

2.5.1.1. Client-Server-Architekturmodell

Beim *Client-Server-Modell* werden die Kommunikationspartner in einem Verteilten System in zwei grundlegende Klassen eingeteilt (siehe Abbildung 2.6(a)). Dies ist zum einen die Klasse der *Server*, die alle Komponenten enthält, die einen bestimmten Dienst anbieten. Diese Dienste werden dabei meist durch langlebige Prozesse implementiert. Dem gegenüber steht die Klasse der *Clients*, die die Komponenten enthält, die jene von den Servern angebotenen Dienste nutzen. Solche Clientprozesse sind im Allgemeinen im Verhältnis zu den Serverprozessen relativ kurzlebig, da sie nur für eine begrenzte Zeit ablaufen, um eine dezidierte Aufgabe zu erfüllen. [TS02, Ham05]

Eine eindeutige Einteilung der Komponenten in eine der beiden Klassen ist dabei nicht immer möglich, da sich Server zur Erfüllung ihres Dienstes selbst wieder als Client an andere Server werden können. Eine Aussage über die Rollenverteilung in einem Client-Server-System kann also immer nur für einen einzelnen Dienstzugriff getroffen werden.

Die Dienstaufrufe im Client-Server-Modell folgen dem *Request-Reply-Protokoll*. Dies bedeutet, dass der Client eine Nachricht mit allen zur Ausführung des Dienstes notwendigen Parametern an einen Server übermittelt, um dessen Dienst anzufragen. Als Antwort erhält der Client daraufhin eine Nachricht mit dem Ergebnis der Dienstausführung. [TS02]

Vorteile des Client-Server-Modells sind die durch die Zentralisierung bedingte leichte Wartbarkeit und die einfachere Gewährleistung der Konsistenz benötigter Daten, da meist keine Kopien verwaltet werden müssen. Zudem kann durch eine Spezialisierung des Servers oftmals eine besonders gute Ausführung seines Dienstes erreicht werden. Dem gegenüber können zentrale Server jedoch schneller zum Flaschenhals werden und leichter von außen attackiert werden. Zudem sind ein Austausch und eine Erweiterung aufgrund der strategischen Position der Server im System meist schwierig und kostspielig. [Sch05, SW05]

2.5.1.2. Peer-to-Peer-Architekturmodell

Das *Peer-to-Peer-Modell* sieht im Gegensatz zum Client-Server-Modell keine dezidierten Rollen vor. Vielmehr sind alle Kommunikationspartner, die so genannten *Peers*, gleichberechtigt und kommen ohne zentrale Kontrolle aus. Die Peers bieten dabei nicht nur Dienste an, sondern nutzen auch die Dienste anderer Peers (vgl. Abbildung 2.6(b)). [Cla01, Ham05]

Peer-to-Peer-Systeme stellen also die Maschine-zu-Maschine-Kommunikation in den Vordergrund, bei der gleichgestellte Partner kooperieren und Ressourcen teilen. Da dieses Modell ohne zentrale Komponenten auskommt, skalieren solche Systeme im Allgemeinen besser als reine Client-Server-Netzwerke und sind zudem robuster gegenüber Ausfällen einzelner Komponenten. [Cla01, Ora01]

Die besonderen Vorteile von Peer-to-Peer-Systemen liegen in ihrer vollständig dezentralen Architektur und Ressourcennutzung sowie in ihrer Fähigkeit zur Selbstorganisation. Dies bedeutet, dass es bei der Nutzung von Diensten weniger häufig zu Engpässen kommt, die Systeme weniger anfällig für böswillige Angriffe (*Hacking*) sind und zudem besser skalieren können. In Peer-to-Peer-Systemen müssen jedoch die Teilnehmer dazu motiviert werden, nicht nur Dienste zu konsumieren, sondern selber auch Dienste bereit zu stellen, um eine Balance zwischen Kosten und Nutzen für den Nutzer zu finden. Zudem ist für die Akzeptanz eines solchen Systems das Vertrauen unter den Teilnehmern ein kritischer Faktor, gerade wenn mit persönlichen oder kritischen Daten umgegangen wird. [Sch05, SW05]

2.5.2. Von Verteilungstransparenz zur *Awareness*

Verteilungstransparenz ist in traditionellen Verteilten Systemen eine der zentralen Eigenschaften, um die Nutzung des Systems für Benutzer und Anwendungen möglichst einfach und optimal zu gestalten (vgl. Abschnitt 2.3.1). Jedoch sind die hierfür entwickelten Technologien meistens für stationäre Verteilte Systeme entworfen worden und können deshalb nicht uneingeschränkt für mobile Umgebungen übernommen werden. So werden zum Beispiel häufig zuverlässige Verbindungen mit hohen Bandbreiten angenommen, wogegen mobile Systeme jedoch meist nur unzuverlässige Verbindungen mit geringen Bandbreiten bieten. Zudem entsteht durch die intrinsischen Eigenschaften mobiler Systeme der Bedarf, Anwendungen dynamisch an Veränderungen in der Umgebung anzupassen (vgl. Abschnitt 2.4). Diese Notwendigkeit zur Anpassung mobiler Systeme an ihren sich ändernden Ausführungskontext wird zusätzlich durch ein häufig vorhan-

denes Ungleichgewicht von zur Verfügung stehenden und benötigten Ressourcen verstärkt. Dies können dabei elementare Ressourcen wie Energie oder eine benötigte Kommunikationsgüte sein oder aber Ressourcen auf höherer Ebene wie Dienste anderer Systemkomponenten oder Benutzerinteraktionen. [AGRS05, Sat04, CEM01, Sat96]

Diesen Bedarf im Bereich mobiler Systeme zu ignorieren und einfach existierende Middleware-Technologien für traditionelle Verteilte Systeme einzusetzen, kann zu einer schlechten Performanz bis hin zur Unbenutzbarkeit des Systems führen, erhöhte Kosten verursachen und zudem die Skalierbarkeit einschränken. Um dies zu vermeiden, müssen sich mobile Systeme hingegen ihrer Mobilität bewusst werden (*Awareness*), um ihr Verhalten an den sich ändernden Ausführungskontext anpassen zu können (*Adaptability*). Dazu muss eine Middleware für mobile Umgebungen Meta-Daten über die Anwendung selbst und ihren Ausführungskontext bereithalten. Dies können relativ einfache Daten wie die Display-Größe oder der verfügbare Speicher sein, aber auch abstrakte Informationen wie die Situation, in der sich ein Nutzer gerade befindet. [Sat04, CEM01]

Die Integration der Awareness in mobile Systeme bedeutet jedoch nicht, dass alle Aspekte der Verteilungstransparenz (vgl. Tabelle 2.1) vollständig aufgegeben werden. Im Gegenteil, denn auch auf diese so genannten Kontextinformationen sollte möglichst einheitlich und somit transparent zugegriffen werden können.

2.5.3. Drahtlose Vernetzung als Grundlage mobiler Systeme

Die Vernetzung mobiler Systeme mit ihren dynamischen Kommunikationsbeziehungen basiert meist auf drahtlosen Kommunikationstechniken. Diese bieten den Teilnehmern innerhalb ihrer Netzabdeckung die Möglichkeit, sich frei zu bewegen, ohne dabei die Fähigkeit zur Kommunikation zu verlieren. Dieser Bereich, in dem eine Übertragung stattfinden kann, ist jedoch begrenzt, da die Qualität der Verbindung mit zunehmender Distanz zwischen den Kommunikationspartnern durch Dämpfung des Signals beeinträchtigt wird. Hierbei gilt, selbst unter optimalen Kommunikationsbedingungen im Vakuum, dass die empfangene Leistung zwischen Sender (P_r) und Empfänger (P_0) quadratisch zur Distanz (r) abnimmt (vgl. Gleichung 2.1) [Sch03].

$$P_r \approx \frac{P_0}{r^2} \tag{2.1}$$

Durch die Dämpfung ist also der Übertragungsbereich eines Senders abgegrenzt. An diesen auch als Funkzelle bezeichneten Bereich schließt sich der

Erkennungsbereich an, in dem zwar erkannt wird, dass ein Sender Signale aussendet, diese aber nicht mehr korrekt interpretiert werden können. Außerhalb des Erkennungsbereichs folgt der Interferenzbereich, in dem zwar das Signal des Senders nicht mehr erkannt wird, dieses jedoch noch Übertragungen anderer Sender stören kann [Sch03].

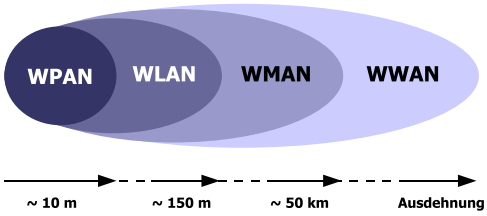


Abbildung 2.7.: Taxonomie drahtloser Netzwerke (nach (CCL03))

Je nach Größe des Übertragungsbereichs werden dabei *Wide und Metropolitan Area Networks* (mit überregionaler Netzabdeckung), *Local Area Networks* (mit lokaler Reichweite) und *Personal Area Networks* (mit einer Reichweite nur in unmittelbarer Umgebung des Senders) unterschieden (vgl. Abbildung 2.7). Der typische Aufbau dieser Netzwerktypen und ihre Einsatzmöglichkeiten im Mobile Computing soll im Folgenden untersucht werden.

2.5.3.1. Wireless Personal Area Networks

Wireless Personal Area Networks (WPAN) verbinden drahtlos Geräte innerhalb eines sehr begrenzten und persönlichen Bereichs, dem *Personal Operating Space* (POS). Das Hauptziel von WPANs ist es, eine kabelgebundene Vernetzung von sich räumlich sehr nahen Geräten zu ersetzen. Dabei wird besonderer Wert darauf gelegt, dass eine Vielzahl von unterschiedlichen Geräten miteinander verbunden werden kann, auch dann, wenn ihre Leistungsfähigkeit begrenzt ist. [SGBH00]

Die IEEE-Arbeitsgruppe 802.15[2] für WPANs definiert dieses persönliche Betriebsumfeld als den Bereich von ca. zehn Metern um eine Person, in dem Geräte dieser Person miteinander kommunizieren. Diese auch als Pikonetze bezeichneten POSs unterstützen im Besonderen das Bilden von Ad-hoc-Netzwerken und dienen zudem als Brücke in andere Netzwerke, wie LAN und WAN [Sch03].

Zurzeit ist die bekannteste Technologien zum Aufbau von WPANs der Bluetooth-Protokollstapel (IEEE 802.15.1), der derzeit in der Version 2.0

[2] http://www.ieee802.org/15/

eine Übertragungsrate von bis zu 3 MBit pro Sekunde bietet und im Folgenden kurz erläutert wird. Daneben werden aber auch Standards für Hochgeschwindigkeits-WPANs mit Übertragungsraten von über 20 MBit pro Sekunde (IEEE 802.15.3) und ressourcensparende WPANs mit deutlich geringeren Datenraten (IEEE 802.15.4) entwickelt.

Bluetooth-Topologie Die Bluetooth-Technologie dient zum Aufbau von Ad-hoc-Pikonetzen, die ohne zusätzliche Infrastruktur auskommen und eine nur relativ geringe Ausdehnung haben. Die Geräte innerhalb eines Bluetooth-Netzes senden ihre Daten über 79 Kanäle im lizenzfreien 2,4-GHz-Band mit einem Kanalabstand von einem MHz und einer Symbolrate von einer Million Symbole pro Sekunde. Je nach Kodierung der Symbole können so zurzeit Datenraten zwischen ein und drei MBit pro Sekunde erreicht werden. Dabei wird die Sendefrequenz mit 1600 pseudo-zufälligen Sprüngen pro Sekunde gewechselt, um Kollisionen mit anderen Sendern im gleichen Frequenzband zu minimieren. [Blu04, Sch03]

Alle Bluetooth-Verbindungen zwischen zwei Geräten werden im Rahmen eines eigenständigen Pikonetzes realisiert (vgl. Abbildung 2.8). Ein solches Pikonetz besteht dabei aus mindestens zwei oder aber mehreren Geräten, die sich einen Frequenzkanal teilen und auf die gleiche Sprungfolge zwischen den Kanälen synchronisiert sind. Innerhalb des Pikonetzes übernimmt ein Gerät die Funktion einer *Leitstation (Master)*, durch deren Uhr und Adresse auch für alle *Folgestationen (Slave)* der jeweils nächste Frequenzwechsel bestimmt wird. Dabei kann eine Station immer nur in einem einzigen Pikonetz als Master fungieren. Als Slave können die Stationen jedoch auch in mehreren Pikonetzen aktiv sein, indem sie im Zeitmultiplex-Verfahren zwischen den Frequenzkanälen der verschiedenen Netze wechseln. In einem solchen Fall spricht man auch davon, dass die Station an einem *Streunetz (Scatternet)* teilnimmt. [Blu04, Sch03]

Aktiv können maximal acht Geräte an solch einem Pikonetz teilnehmen, da der Adressbereich lediglich drei Bit breit ist. Als passive, *geparkte Mitglieder* können sich zusätzlich weitere 256 Stationen im Pikonetz befinden. Bei Bedarf können diese innerhalb weniger Millisekunden wiederbelebt und somit aktiv werden. Stationen, die sich lediglich im *Bereitschaftsmodus (Standby)* befinden, können erst nach Beitritt zum Pikonetz an der Kommunikation teilnehmen. Dabei ist es innerhalb des Bluetooth-Netzes möglich, Daten *synchron*, *asynchron* und *isochron* zu übertragen. Zudem wird die Option der Datenübermittlung per *Broadcast* unterstützt. [Blu04, Sch03]

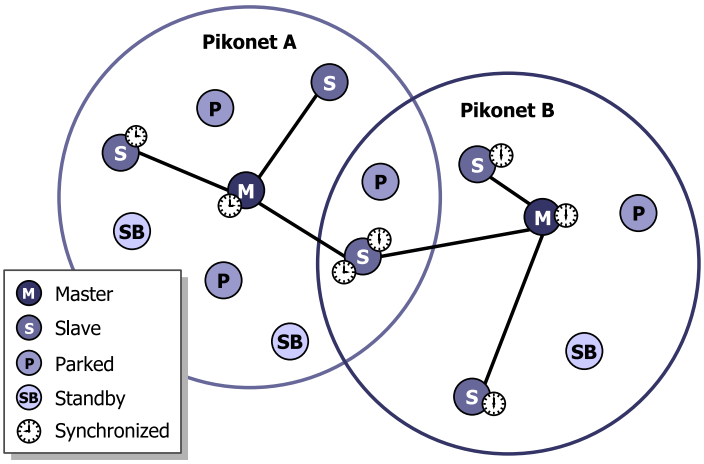


Abbildung 2.8.: Bluetooth-Streunetze (nach (Sch03))

Bluetooth-Architektur Der Aufbau des Bluetooth-Protokollstapels kann grob in eine *Kernspezifikation* und in eine *Profilspezifikation* unterteilt werden. Der Kern der Architektur wird vom *Bluetooth Controller*, dem *Logical Link Control and Adaption Protocol* (L2CAP) und dem *Service Discovery Protocol* (SDP) gebildet. Die Profilspezifikation erweitert diese Protokolle und Spezifikationen um ergänzende Funktionen und Protokolle zur Anpassung der Bluetooth-Funktechnik an bestehende sowie zukünftige Anwendungen und Technologien (siehe Abbildung 2.9). [Blu04, Sch03]

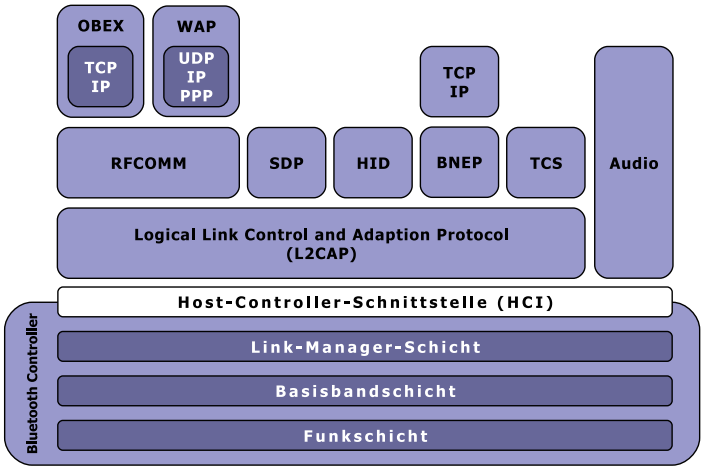


Abbildung 2.9.: Der Bluetooth-Protokollstapel (nach (Blu04, HA03))

Der Mediumzugriff wird mittels des Bluetooth Controllers mit seiner *Funk-*, *Basisband-* und *Link-Manager-Schicht* realisiert. Auf diese Architekturkomponente, die in etwa die physikalische Schicht und Teile der Sicherungsschicht des ISO/OSI Referenzmodells umfasst, wird über die

Host-Controller-Schnittstelle (HCI) zugegriffen, um das Bluetooth-System zu steuern und Daten zu versenden. Besonders interessant für den Einsatz im Mobile Computing ist, dass das Erkennen von Bluetooth-Stationen in Funkreichweite bereits in diese zentrale Komponente integriert ist.

Ergänzt wird die Sicherungsschicht durch die *L2CAP-Schicht*. Sie bietet höheren Schichten einen asynchronen und kanalbasierten Datentransfer mit eigener Fehlerkontrolle. Dabei wird zwischen drei Kanalarten unterschieden: *Verbindungslose Kanäle*, die unidirektional Stationen miteinander verbinden und meist für Broadcasts genutzt werden. Daneben gibt es *verbindungsorientierte Kanäle*, die bidirektionalen Datentransfer erlauben, und *Signalisierungskanäle* zum Austausch von Signalnachrichten zwischen L2CAP-Instanzen. Der Zugriff auf den vom Controller bereitgestellten synchronen Datentransfer, wie er zum Beispiel für Audioanwendungen benötigt wird, wird durch L2CAP nicht unterstützt und muss deshalb direkt über das HCI abgewickelt werden (vgl. Abbildung 2.9). [Blu04, Sch03]

Das Kernsystem wird durch das *Service Discovery Protocol (SDP)* komplettiert. Dabei ist es für eine Umgebung konzipiert, in der sich die Anzahl verfügbarer Dienste dynamisch ändert. Mit Hilfe von SDP ist es möglich, inkrementell Informationen über beliebige, aber eindeutig identifizierbare Dienste in der Bluetooth-Umgebung zu erlangen. Das Protokoll ist so gestaltet, dass es dienstunabhängig ist, ohne zentrale Komponenten auskommt (*Peer-to-Peer-Ansatz*) und das Erscheinen neuer Dienste sowie den Wegfall bekannter Dienste erkennt. [Blu04]

Die Profilspezifikation, die auf dem Kernsystem aufbaut, dient zum einen dazu, klassische kabelgebundene Kommunikationsformen zu ersetzen und zum anderen bestehende Protokolle zu adaptieren (vgl. Abbildung 2.9). So emuliert zum Beispiel das RFCOMM-Protokoll eine serielle Schnittstelle nach EIA[3]-232 (ehemals RS-232). Eingabegeräte, wie Tastaturen oder Mäuse, können mit Hilfe des *Human Interface Device Protocol* drahtlos an bluetoothfähige Geräte angebunden werden. Zudem können Internetanwendungen wie gewohnt den TCP/IP-Protokollstapel verwenden, wobei diese Protokolle jedoch mit Hilfe von RFCOMM oder dem *Bluetooth Network Encapsulation Protocol (BNEP)* an den Bluetooth-Protokollstapel angepasst werden. Aber auch eigene Protokolle und Dienste lassen sich auf Basis des Kernsystems beziehungsweise bestehender Protokolle der Profilspezifikation einfach anbieten und betreiben. [Sch03, HA03]

Die Bluetooth-Architektur bietet insgesamt eine sehr gute Basis für Mo-

[3] http://www.eia.org/

bile-Computing-Systeme. Dies begründet sich vor allem darauf, dass wesentliche Aspekte, wie das Auffinden von Geräten und Diensten in der mobilen Umgebung, bereits vom Kernsystem dienstunabhängig bereitgestellt werden.

2.5.3.2. Wireless Local Area Networks (WLAN)

Drahtlose lokale Netze (WLAN) decken im Gegensatz zu Personal Area Networks (vgl. 2.5.3.1) nicht nur den unmittelbaren Arbeitsbereich einer Person ab, sondern bieten mit einer durchschnittlichen Reichweite von ungefähr 150 Metern [Wis05] die Möglichkeit, ganze Etagen beziehungsweise Gebäude drahtlos zu vernetzen. Dabei ist auch hier eines der Hauptziele, die eher unflexible kabelgebundene Vernetzung zwischen Rechnern, Infrastrukturkomponenten und externen Netzen (wie zum Beispiel dem Internet) zu ersetzen. Dabei haben WLANs vor allem Vorteile in Bezug auf *Flexibilität*, *einfache Planung*, *Robustheit* und *Kosten*. Demgegenüber stehen jedoch auch Nachteile bezüglich *Dienstgüte*, *Heterogenität der Techniken* und im besonderen Maße in punkto *Sicherheit*. [Sch03, Wis05]

Im Detail zeigen sich die Vorteile darin, dass durch den Einsatz von WLANs eine zerstörungsfreie Vernetzung auch durch Wände hinweg geschehen kann, die beteiligten Netzknoten innerhalb des Versorgungsbereichs mobil sind und so auch spontane Veränderungen bezüglich Anzahl und Ort der Netzkomponenten leicht realisiert werden können. Das Fehlen von Kabeln reduziert zudem die Installationskosten und verhindert Netzausfälle, die aufgrund defekter oder zerstörter Leitungen auftreten. [Sch03, Wis05]

Hiergegen abzuwägen ist die im Verhältnis zu kabelgebundenen Lösungen praktisch in jeglicher Hinsicht schlechtere Dienstgüte, wie zum Beispiel eine geringere maximale Nutzdatenrate oder eine höhere Fehlerhäufigkeit. Zudem gibt es historisch- und regulatorisch bedingt eine Vielzahl von konkurrierenden Technologien, die sich aber zunehmend konsolidieren. Der wohl gravierendste Nachteil ist jedoch, dass es sich bei der drahtlosen Datenübertragung im Allgemeinen um eine Broadcast-Kommunikation handelt, bei der jeder innerhalb des Übertragungsbereichs, ob erlaubt oder nicht, die versendeten Daten empfangen und auswerten kann. Deshalb sind bei der Nutzung von WLAN zusätzliche Sicherheitstechnologien, wie zum Beispiel die Verschlüsselung der Nutzdaten oder die Authentifizierung beteiligter Netzknoten, meistens unabdingbar. [Sch03]

Als einer der bekanntesten Vertreter drahtloser Netzwerke gilt der

IEEE-Standard 802.11[4], der in verschiedenen Versionen existiert und Datenraten von ein beziehungsweise zwei MBit pro Sekunde in der Ursprungsversion bis hin zu momentanen 540 MBit (802.11n[5]) pro Sekunde bietet.

IEEE-802.11-Architektur Der von der IEEE entwickelte WLAN-Standard zum Aufbau drahtloser Netzwerke wird wie auch die Ethernet-, Token-Bus- oder Token-Ring-Standards vom Komitee 802 für lokale Netzwerke normiert und fügt sich nahtlos in diese Familie ein. Deshalb umfassen die WLAN-spezifischen Erweiterungen hauptsächlich die Verwaltung der Luftschnittstelle auf der Medium-Access-Control-Schicht (MAC-Schicht) sowie die elementare Kodierung und Übertragung der Daten auf der physikalischen Schicht (siehe Abbildung 2.10).

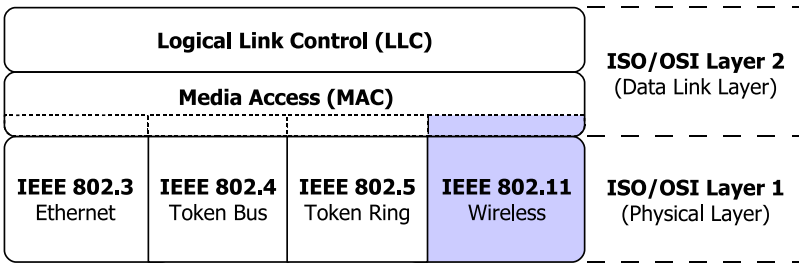


Abbildung 2.10.: Einordnung des WLAN-Standards in die IEEE-802-Familie

Hierbei wird auf der MAC-Schicht der Zugriff der Netzwerkknoten auf das gemeinsame Medium mit Hilfe des CSMA/CA[6]-Verfahrens gesteuert und zu große Dateneinheiten fragmentiert beziehungsweise wieder zusammengefügt. Die darunter liegende physikalische Schicht gliedert sich intern in zwei weitere Schichten: zum einen in die *Physical-Layer-Convergence-Procedure*-Subschicht (PLCP) und zum anderen in das *Physical-Medium-Dependent*-System (PMD). Dabei hat die PLCP-Schicht die Aufgabe, die Dateneinheiten der MAC-Schicht so anzupassen, dass sie zwischen zwei oder mehreren Netzwerkknoten mit Hilfe der PMD-Schicht ausgetauscht werden können. Diese definiert abschließend die Charakteristika und Methoden, um Daten drahtlos zu übermitteln und zu empfangen. [IEE99]

[4] http://www.ieee802.org/11/
[5] Eine endgültige Verabschiedung des Standards ist bis Juni 2009 angestrebt.
[6] Carrier Sense Multiple Access / Collision Avoidance

IEEE-802.11-Topologie Der IEEE-802.11-Standard kann grob in zwei unterschiedliche Funktionsmodi eingeteilt werden, die sich auf die Art der Kommunikationsbeziehung zwischen den Stationen des Netzwerkes beziehen. Dies ist einerseits die Kommunikation im *Infrastrukturmodus* (vgl. Abschnitt 2.3.2.2 und andererseits die direkte Übertragung zwischen den Stationen im *Ad-hoc-Modus* (vgl. Abschnitt 2.3.2.1).

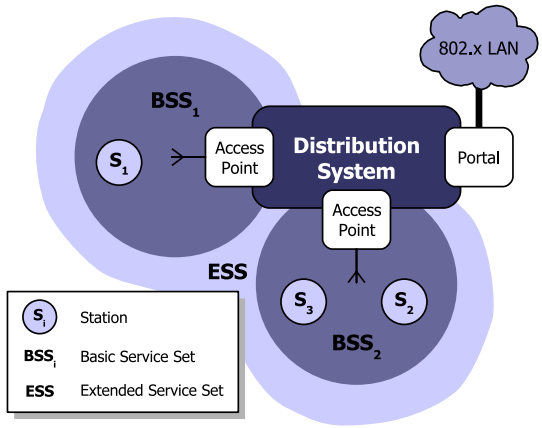


Abbildung 2.11.: Architektur eines infrastrukturbasierten IEEE-802.11-Netzwerks (nach (Sch03))

Im Infrastrukturmodus (siehe Abbildung 2.11) fungiert ein *Access Point* (AP), auch *Basisstation* genannt, als Bridge beziehungsweise Router und formt mit den ihm drahtlos verbundenen Netzwerkstationen (S) ein *Basic Service Set* (BSS). Alle Stationen, die sich in demselben BSS befinden, können miteinander kommunizieren. Dies ist selbst dann möglich, wenn sie keinen direkten Funkkontakt haben, da der Access Point die Daten im Service Set verteilt. Werden mehrere BSS mittels eines Distributionssystems (DS) miteinander verbunden, so entsteht ein logisch größeres Netzwerk, das dann als *Extended Service Set* (ESS) bezeichnet wird. Zwischen den einzelnen Access Points eines ESS ist zudem eine Übergabe mobiler Netzwerkknoten möglich (*Handover*).

Der Aufbau des Distributionssystems wird dabei bis auf die zu erbringende Dienstleistung nicht näher vom Standard spezifiziert. Häufig werden hierfür jedoch, da sich der WLAN-Standard nahtlos in die IEEE-802-Familie eingliedert, andere lokale Netzwerktypen dieser Familie verwendet. Über ein an das DS angeschlossenes Portal kann das drahtlose Netzwerk zudem an weitere Netze, wie zum Beispiel dem Internet, angeschlossen werden. [AGRS05, Sch03]

Ohne Access Point können zwei oder mehrere Stationen, die sich untereinander in Funkreichweite befinden, auch ein Ad-hoc-Netzwerk bilden,

welches als *Independent Basic Service Set* (IBSS) bezeichnet wird. Eine indirekte Kommunikation zwischen zwei Stationen (Routing von Daten) ist im Standard zurzeit nicht vorgesehen, sodass die räumliche Ausdehnung dieser IBSS meist sehr gering bleibt.

Aufgrund der Motivation, einen Standard zu schaffen, der die drahtgebundene Verkabelung ersetzt, berücksichtigt der WLAN-Standard keine spezifischen Anforderungen von mobilen Anwendungen. Vielmehr stellt sich die drahtlose Übertragung der Daten transparent dar und ist, infolge der Kapselung auf MAC-Ebene, nicht von der Übertragung durch andere lokale Netzwerkarten unterscheidbar.

2.5.3.3. Wireless Metropolitan und Wide Area Networks (WMAN/WWAN)

Parameter, wie zum Beispiel die maximal erlaubte Latenzzeit[7] setzen der Ausdehnung von drahtlosen Netzen auf WLAN-Basis ein technisches Limit. Um auch größere Flächen drahtlos zu vernetzen, werden Standards für *Wireless Metropolitan Area Networks* (WMAN) und *Wireless Wide Area Networks* (WWAN) eingesetzt. Dabei wird die Grenze zwischen WMANs und WWANs bei einer ungefähren Ausdehnung von 50 Kilometern gezogen. [Int03]

Drahtlose überregionale (Daten-)Netzwerke sind meistens aus der Motivation heraus entwickelt worden, lokale drahtlose Netzwerke mit anderen Netzwerken (zum Beispiel dem Internet) breitbandig zu verbinden. Darauf aufbauend sind dann nach und nach auch Standards entwickelt worden, die mobile Geräte direkt ansprechen. Aber auch aktuelle Mobilfunknetze, wie zum Beispiel UMTS (*Universal Mobile Telecommunications System*), bieten digitale Dienste mit Datenraten von bis zu 2 MBit pro Sekunde an. Damit sind die drahtlosen überregionalen Netzwerke zu echten Alternativen zum drahtgebundenen Anschluss der „letzten Meile“ mittels Kabelmodem oder DSL (*Digital Subscriber Line*) geworden. [AGRS05, Sch03, Int03]

Mobile WiMAX Mobile WiMAX (*Worldwide Interoperability for Microwave Access*) basiert auf den IEEE-Standards 802.16[8] für drahtlose Breitbandnetze und 802.16e, der die Erweiterung des Standards für mobile Systeme darstellt. Auf dieser Basis definiert das *WiMAX-Forum*[9] System

[7] Der WLAN-Standard erlaubt eine Latenzzeit von ungefähr 900 Nanosekunden.
[8] http://www.ieee802.org/16/
[9] http://www.wimaxforum.org/

Profile, die für abgegrenzte Einsatzgebiete die verpflichtenden und optionalen Eigenschaften aus dem IEEE-Standard definieren, um Interoperabilität zwischen verschiedenen Umsetzungen zu erreichen (vgl. Abbildung 2.12). Darüber hinaus legt es die grundlegende Netzwerkarchitektur fest, die für den Aufbau von mobilen WiMAX-Netzwerken auf Ende-zu-Ende-Ebene notwendig ist. [WiM06, Int03]

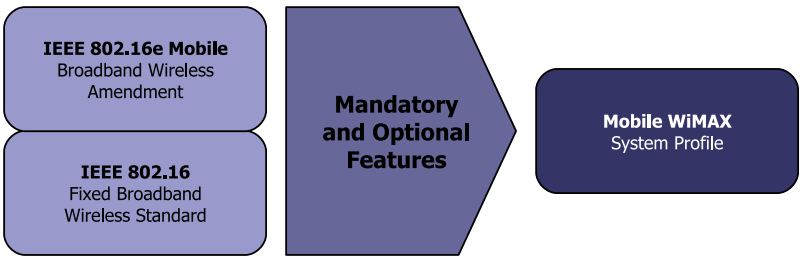


Abbildung 2.12.: Mobile-WiMAX-Systemprofile (nach (WiM06))

Systeme, die entsprechend dem Mobile-WiMAX-Standard entworfen sind, zeichnen sich durch hohe Datenraten aus, die in einem 10-MHz-Kanal bis zu 63 MBit pro Sekunde beim Empfang und bis zu 28 MBit pro Sekunde beim Senden betragen können. Zudem bietet Mobile WiMAX garantierte Dienstgüten (*Quality of Service*) und eine gute *Skalierbarkeit* an, indem die verfügbare Bandbreite in Subkanäle mit variabler Bandbreite geteilt wird und diese dann einzelnen Anwendungen zugeordnet werden. Dies geschieht mit Hilfe des so genannten *Scalable Orthogonal Frequency Division Multiplexing* (S-OFDM). Aber auch Techniken zur Absicherung der drahtlosen Kommunikation, wie Authentifizierung und Verschlüsselung, sind in WiMAX bereits enthalten. [WiM06]

Um den besonderen Eigenschaften mobiler Systeme gerecht zu werden, sind zum einen Mechanismen zum *Energiemanagement* und zum anderen Methoden zum Wechsel zwischen Basisstationen (*Handoff*) enthalten.

So können mobile Stationen sich zum Beispiel für eine ausgehandelte Zeit von einer Basisstation abmelden (*Sleep Mode*) oder nur noch periodisch auf Broadcast-Nachrichten reagieren (*Idle Mode*) und so Energie sparen. Um einen möglichst verzögerungsfreien Übergang von einer Basisstation zur anderen zu gewährleisten, definiert WiMAX im Rahmen der durch IEEE-802.16e vorgegebenen Rahmenbedingungen optimierte Handoff-Methoden, die die Verzögerung auf unter 50 Millisekunden verbessern. [WiM06]

2.6. Sicherheits- und Vertrauensproblematik

Mobile Verteilte Systeme sind grundlegend den gleichen Arten von böswilligen Angriffen ausgesetzt, wie ihre stationären und drahtgebundenen Gegenstücke. Jedoch führen die dynamische und heterogene Struktur einer mobilen Umgebung sowie die Verwendung von Broadcast-Kommunikationsformen zu zusätzlichen Angriffsmöglichkeiten und Herausforderungen. Darüber hinaus führen aktuelle kontextbasierte Systeme (vgl. Abschnitt 3.3), die individuelle Anwendungen und Kommunikationsbeziehungen ermöglichen, zu einer höheren und ubiquitären Verfügbarkeit von persönlichen Daten und damit auch höheren Ansprüchen an den Schutz der Privatsphäre. [CCL03, ADW01]

Das Ziel des Einsatzes von Sicherheits- und Vertrauensmechanismen in mobilen Systemen ist demnach, die Nutzer und Geräte sicher zu authentifizieren, Ressourcen und Daten vor unautorisiertem Zugriff zu schützen, die Kommunikation zweier Partner abzusichern und die Systemintegrität zu gewährleisten. Um dies zu erreichen, müssen auf allen Architektur- und Dienstschichten des Systems entsprechende Maßnahmen getroffen werden. Dabei darf der Datenschutz jedoch nicht auf die vollständige Anonymität der persönlichen Daten reduziert werden, um den Situationen gerecht zu werden, in denen der Nutzer gezielt Daten preisgibt, um für ihn vorteilhafte und erweiterte Dienste in Anspruch nehmen zu können. Deshalb ist ein monolithisches Sicherheitskonzept im Mobile Computing meist nicht vorteilhaft. Es muss vielmehr auf der Basis von Sicherheits- und Anonymisierungsverfahren die Möglichkeit geschaffen werden, verschiedene Vertrauensgrade und -bedürfnisse formulieren und durchsetzen zu können. Damit können dann in unterschiedlichen Situationen gezielt nur angemessene Informationen mit eindeutig identifizierten Diensten und Kommunikationspartnern ausgetauscht werden. [HL04, BCS01]

Diese Ziele in mobilen Ad-hoc-Systemen zu erreichen, ist eine besondere Herausforderung, da hier ohne zentrale und vertrauenswürdige Dritte Partei (*Trusted Third Party*) Vertrauen zwischen den Teilnehmern geschaffen werden muss. Dazu sind neue Vertrauensmetriken notwendig, die auf der Basis von vordefiniertem Vertrauen, Empfehlungen anderer, Risikobewertungen und -analysen sowie früheren Erfahrungen zu lokalen Entscheidungen führen. Aber auch der Nachrichtentransport in Ad-hoc-Systemen mit Multi-Hop-Routing, in denen Nachrichten über zum Teil unbekannte Netzwerkknoten zum Empfänger geleitet werden, macht erweiterte Sicherheitsstrategien notwendig, da eine einfache Absicherung des physika-

lischen Kommunikationskanals nicht mehr ausreicht. [CCC+06]

In mobilen ubiquitären Systemen entsteht jedoch ein Dilemma: Auf der einen Seite bedingt eine situationsabhängige Kontrolle von sensiblen Daten eine erhöhte Interaktion mit dem Nutzer, da er zum Beispiel Sicherheitspräferenzen und -regeln definieren muss. Andererseits sollen ubiquitäre Systeme jedoch möglichst eigenständig sowie unauffällig arbeiten und so in den Hintergrund treten (vgl. Einleitung des Abschnitts 2). Dieser Zwiespalt führt dazu, dass Sicherheits- und Vertrauensmechanismen für mobile Systeme selbst kontextsensitiv werden sollten. Auf diese Weise erhalten sie die Möglichkeit, sich der aktuellen Situation möglichst optimal anzupassen und dadurch die notwendige Interaktion mit dem Nutzer zu minimieren. [CCC+06, ADW01]

2.7. Implikationen für diese Arbeit

Das vorstehende Kapitel hat gezeigt, dass durch die Miniaturisierung der Systemkomponenten und die zunehmende Vernetzung digitaler Geräte die allgegenwärtige Verfügbarkeit von Computern stetig steigt und so der globale ubiquitäre Computer als Zukunftsmodell immer realer wird. Dabei wird dieser Trend durch die steigende Konvergenz der zuvor getrennten Daten-, Sprach- und Multimedia-Netze zu Universalnetzen noch begünstigt. Der ubiquitären Zugriff auf Hard- und Software-Ressourcen führt dazu, dass die Mobilität von Geräten, Nutzern und Anwendungen eine zentrale Rolle einnimmt und somit das Gebiet des Mobile Computings hoch aktuell ist.

Mobile Systeme sind zwar Verteilte Systeme, aber die Mobilität bringt eigene Anforderungen, Einschränkungen und Grundlagen mit sich, die eine einfache Adaption traditioneller Methoden und Techniken Verteilter Systeme meist nicht erlaubt. Dabei gilt in der Regel, dass je mehr mobile Teilnehmer an einem System beteiligt sind, seine Dynamik und Heterogenität ebenfalls wächst. Dies führt dazu, dass der Ausführungskontext der Systemkomponenten stetigen Änderungen in Form von verfügbaren Diensten, Ressourcen und Qualitätsparametern unterliegt. Hierbei können mobile Ad-hoc-Systeme, die keine feste Infrastruktur und nur unzuverlässige Kommunikationsmöglichkeiten haben, dafür aber ein gewisses Maß an Selbstorganisation besitzen, als „*Worst Case*“ eines mobilen Systems angesehen werden.

Die intrinsischen Eigenschaften, wie eine erhöhte Ressourcenknappheit, eine oftmals variierende Konnektivität und die große Heterogenität der

beteiligten Systeme und Netzwerke, führen dazu, dass Verteilungstransparenz nur noch eingeschränkt gefordert werden kann. Vielmehr werden sich die mobilen Systeme durch Konzepte wie Awareness und Adaptability ihrer Mobilität bewusst und können sich dadurch möglichst optimal an die Veränderungen im Ausführungskontext anpassen.

Mobile System stellen aber auch erhöhte Anforderungen an die Sicherheit, da sie differenzierte Sicherheits- und Vertrauensgrade benötigen, um Awareness und Adaptability überhaupt erst zu ermöglichen. Hierbei muss immer zwischen dem erwarteten zusätzlichen Nutzen und den daraus entstehenden Risiken abgewogen werden, um möglichst optimal die Ressourcen des Gesamtsystems nutzen zu können.

Das nächste Kapitel befasst sich deshalb mit Middleware-Plattformen, welche die Konstruktion und die Ausführung verteilter Anwendungen in mobilen Umgebungen mit ihren speziellen Rahmenbedingungen unterstützen. Dabei werden die erweiterten Anforderungen mobiler Middleware-Systeme aufgezeigt sowie aktuelle Konzepte und Plattformen, die solche Systeme realisieren, eingeführt. Das Kapitel bildet damit die Basis, um bestehende Defizite zu verdeutlichen und ein Konzept zur benutzerzentrierten Erweiterung zu entwickeln.

3. Middleware-Plattformen zum Aufbau mobiler Umgebungen

Mobile Verteilte Systeme werden von Anwendungen vor allem als Kommunikationsinfrastruktur zum Zugriff auf entfernte Daten, Dienste und Komponenten verwendet (vgl. Abschnitt 2.5). Dazu können die Anwendungen zunächst nur direkt auf die vom System angebotenen Netzwerkschnittstellen zugreifen, die Basisdienste, wie Verbindungsauf- und -abbau, die Übertragung von Byte-Strömen und eine grundlegende Fehlerbehandlung bieten (*Netzwerkprogrammierung*). Komplexere Anforderungen an die Kommunikation müssen dann jedoch von den Anwendungen selbst implementiert werden. Dabei verlieren die Systeme aber im Allgemeinen ihre Offenheit und Transparenz, können aber an Einfachheit und Performanz gewinnen. [Ham05, TS02]

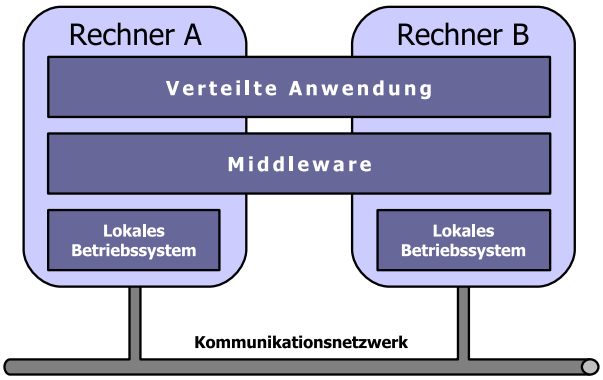


Abbildung 3.1.: Ein Verteiltes System mit Middleware-Schicht (nach (TS02))

Um die Notwendigkeit der direkten Netzwerkprogrammierung zu minimieren und zusätzlich die Heterogenität der verschiedenen Netzwerkanbindungen, Hardware, Betriebssysteme und Programmiersprachen zu maskieren, wird eine zusätzliche Software-Schicht, die so genannte *Middleware*, zwischen Anwendung und lokalem Betriebssystem eingefügt (siehe Abbildung 3.1). Diese Schicht soll die Verteilungstransparenz verbessern und die Anwendung in ihren Abläufen unterstützen, indem einheitliche Modelle für den Zugriff auf Dienste und Datenobjekte sowie erweiterte Basisdienste, wie zum Beispiel eine Transaktionsunterstützung, bereitgestellt werden. Je nachdem, wie stark eine Middleware-Plattform

Anwendungen über die reine Abstraktion der Netzwerkprogrammierung unterstützt, können Middleware-Systeme in zwei Kategorien eingeteilt werden: Dies ist zum einen die Klasse der *kommunikationsorientierten Middleware*, die sich eher auf die Maskierung des Netzwerkzugriffs beschränkt und zum anderen die Klasse der *anwendungsorientierten Middleware*, deren Vertreter zusätzlich zur Kommunikation auch Dienste für eine weitergehende Unterstützung der mobilen Anwendung bereitstellen. [Ham05, TS02, CDK01]

In diesem Kapitel werden zunächst grundlegende Anforderungen und Eigenschaften von Middleware-Plattformen im Mobile Computing erörtert. Danach werden wichtige Middleware-Konzepte im Bereich mobiler Verteilter Systeme eingeführt und exemplarische Vertreter der einzelnen Konzeptklassen erläutert.

3.1. Anforderungen an Middleware-Systeme im Mobile Computing

So wie mobile Verteilte Systeme grundlegende Eigenschaften traditioneller verteilter Systeme besitzen (vgl. Abschnitt 2.3.2), können auch Middleware-Systeme für das Mobile und Ubiquitous Computing als Weiterentwicklung traditioneller Middleware-Systeme angesehen werden. Sie umfassen dabei allgemeine Basisaufgaben, die nicht zu den Kernaufgaben der verschiedenen Anwendungen gezählt werden können. Eine Middleware bildet somit eine höhere Abstraktionsebene, die – im Gegensatz zur lokalen Ressourcenverwaltung des Betriebssystems – die Ressourcen im Verteilten System verwaltet und einheitlich zugreifbar macht. Dabei ist eine exakte Abgrenzung zwischen Betriebssystem-, Middleware- und Anwendungsschicht nicht immer möglich, sodass einzelne Funktionalitäten manchmal nicht eindeutig einer Schicht zugeordnet werden können. [Sch05, TS02, MCE02]

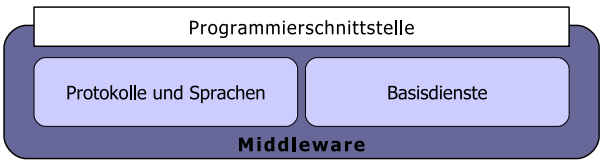


Abbildung 3.2.: Komponenten einer Middleware (nach (Sch05, TS02))

Der Aufbau einer Middleware kann in drei Komponenten gegliedert werden, die die Funktionalität der Middleware bestimmen (vgl. Abbildung 3.2). Dies ist zum einen eine *Programmierschnittstelle*, die einen einheit-

lichen Zugriff auf die Dienste und Komponenten der Middleware auf den verschiedenen Systemen gewährleistet. Zum anderen definiert eine Middleware verpflichtende und optionale *Basisdienste* sowie *Sprachen und Protokolle* zur Kommunikation zwischen den Middleware-Instanzen der einzelnen (mobilen) Systeme. [Sch05, TS02]

Basisdienste Basisdienste aggregieren allgemeine Funktionalitäten, die nicht direkt zu den Kernaufgaben von Anwendungen oder Betriebssystemen gehören und können vielfältige administrative Aufgaben umfassen. Zu diesen Diensten werden zum Beispiel das Auflösen von logischen Namen in physikalische Adressen oder das Registrieren und Auffinden von Diensten gezählt. Aber auch Abrechnungsfunktionalitäten, das Bereitstellen von Sicherheitsfunktionalitäten oder die Garantie von Dienstgüteparametern (QoS) zählen zu ihnen. [Sch05]

Protokolle und Sprachen Protokolle definieren das Regelgefüge, mit dem in einem Verteilten System strukturiert kommuniziert wird. Dabei geben sie das Format, die Bedeutung und die Reihenfolge der innerhalb einer Kommunikationsschicht ausgetauschten Nachrichten an. Protokolle dienen also zur Realisierung des Dienstes einer Systemkomponente. Da sie jedoch relativ starr den Nachrichtenaustausch definieren, können komplexere Inhalte meist nicht direkt mit ihnen ausgedrückt werden. Hierzu können dann Sprachen eingesetzt werden, die über eine Grammatik und ein festes Vokabular spezifiziert sind. Dabei steht jedoch der größeren Flexibilität meist ein höherer Verarbeitungsaufwand entgegen. Verallgemeinert kann man sagen, dass mit den Sprachen die Daten beschrieben werden, die dann mit Hilfe der Protokolle übertragen werden. Beispielhaft hierfür sind die Dokumente im *World Wide Web*, die in der Auszeichnungssprache *(X)HTML* geschrieben und über *HTTP* zwischen Web Client und Web Server ausgetauscht werden. [Sch05, Tan97]

Anforderungskatalog Middleware-Systeme im Mobile Computing haben zwar ähnliche Basisanforderungen, besitzen aber zusätzliche Anforderungen, die sich aus den Eigenschaften mobiler Verteilter Systeme ableiten (vgl. Abschnitt 2.3). So können vor allem keine stabilen und konstanten Kommunikationsverbindungen und Umgebungseigenschaften (Kontextinformationen) vorausgesetzt werden. Zudem muss die meist vorhandene Ressourcenarmut der mobilen Geräte berücksichtigt werden. Zusammenfassend sollte eine Middleware im Mobile Computing folgende allgemeine

und spezifische Anforderungen erfüllen (vgl. Tabelle 3.1): [Sch05, MCE02]

Eine der wichtigsten allgemeinen Anforderungen an Verteilte Systeme und damit auch an Middleware-Systeme im Mobile Computing ist die gemeinsame Nutzung von Ressourcen (*Ressourcenteilung*). Dies birgt zum einen den Vorteil, dass Ressourcen effizienter genutzt werden können und zum anderen, dass damit die Investition in kostspielige Ressourcen – wie zum Beispiel Supercomputer – oftmals erst sinnvoll ist. Dabei sollte das System eine größtmögliche *Offenheit* bieten. Dies bedeutet, dass es durch die Definition und Verwendung von einheitlichen Schnittstellen und Protokollen relativ leicht ist, neue und heterogene Einheiten in das System zu integrieren. Hinzu kommt, dass Middleware-Systeme möglichst auch bei signifikant steigender Anzahl von Komponenten leistungsfähig bleiben sollen (*Skalierbarkeit*). Wichtig ist auch, dass die Middleware in der Regel alle Verteilungsaspekte, die für die darauf aufbauenden Anwendungen nicht relevant sind, maskieren beziehungsweise verbergen sollen (*Transparenz*). Aber auch eine Toleranz gegenüber dem Ausfall einzelner Komponenten sollte vorhanden sein (*Fehlertoleranz*). Dies bedeutet, dass Teile des Systems ausfallen können und dies andere Teile nicht beeinträchtigt. [Sch05, TS02, CDK01]

Spezifisch für Middleware-Systeme im Mobile Computing ist die Forderung nach einem Bewusstsein über ihre eigene Mobilität (*Awareness*). Diese Anforderung begründet sich aus dem sehr viel dynamischeren Ausführungskontext mobiler Systeme, der es erforderlich macht, dass Umgebungsinformationen den Anwendungen einheitlich bereit gestellt werden, damit diese sich den veränderten Bedingungen optimal anpassen können. Zudem bedingen die meist vorhandenen Ressourcenbeschränkungen mobiler Geräte ein möglichst leichtgewichtiges Systemdesign, um eine ausgewogene Balance zwischen gewünschter Leistung der Middleware und der damit erzeugten Rechenlast zu erhalten (*Leichtgewichtigkeit*). Zudem müssen sowohl die *logische Mobilität* von Softwareeinheiten als auch die *physikalische Mobilität* von gegenständlichen Objekten unterstützt werden. Somit ist es dann möglich, Berechnungen direkt vor Ort auszuführen oder ortsbezogene Dienste anzubieten. Eine Middleware sollte es Anwendungen auch erlauben, für sie relevante Weltausschnitte und Kontextinformationen zu definieren, damit die Kommunikation zwischen Anwendungen desselben Typs standardisiert werden kann (*Domänenbildung*). Dabei kann ein Informationsaustausch nur dann optimal betrieben wer-

Anforderung	Beschreibung
Ressourcenteilung	Durch die geteilte Nutzung der Ressourcen können diese effizient genutzt werden.
Offenheit	Das dynamische Hinzufügen und Entfernen neuer Einheiten sollte durch einheitliche Schnittstellen und Protokolle unterstützt werden.
Skalierbarkeit	Die Middleware sollte auch bei wachsenden Anforderungen leistungsfähig bleiben.
Transparenz	Irrelevante Aspekte der Verteilung sollten verborgen werden.
Fehlertoleranz	Der Ausfall einzelner Komponenten im System sollte nicht zur Beeinträchtigung des Gesamtsystems führen.
Awareness	Monitoring der Umgebung des mobilen Gerätes und Bereitstellen dieser Informationen, um auf Veränderungen reagieren zu können.
Leichtgewichtigkeit	Die eingesetzten Komponenten sollten leichtgewichtig sein, da mobile Geräte meist ressourcenarm sind.
Logische Mobilität	Die Migration von Software-Komponenten sollte unterstützt werden, um zeitnah vor Ort Berechnungen durchführen zu können.
Physikalische Mobilität	Die Mobilität physikalischer Objekte sollte unterstützt werden, um ortsbasierte Dienste zu ermöglichen.
Domänenbildung	Modellierung von relevanten Weltausschnitten, um die Inhalte der Kommunikation zu standardisieren.
Kommunikation	Anbieten von einheitlichen (asynchronen) Kommunikationsverbindungen.
Kompatibilität	Auch klassische Middleware-Systeme und nichtmobile Hard- und Software sollten zur Vermeidung von Redundanzen und zusätzlichem Entwicklungsaufwand unterstützt werden.
Energieeffizienz	Um die Energieressourcen der mobilen Geräte zu schonen, sollten Mechanismen zur effizienten Energieverwaltung in die Middleware integriert werden.
Sicherheit und Vertrauen	Hoch dynamische Umgebungen und ad-hoc entstehende Beziehungen bedingen zusätzliche Sicherheits- und Vertrauensmaßnahmen.

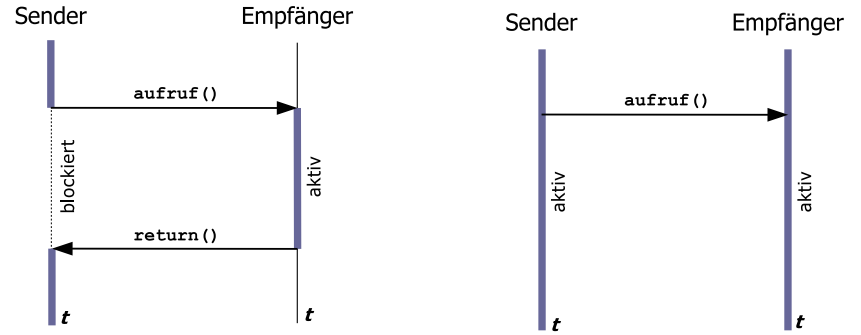
Tabelle 3.1.: Anforderungen an Middleware-Systeme im Mobile Computing

den, wenn die Middleware einen von konkreten Netzwerkprotokollen und -arten unabhängigen Kommunikationsmechanismus bereitstellt, der auf asynchronen Paradigmen aufbaut, um so auch eine *Kommunikation* zwischen temporär nicht erreichbaren Partnern zu erlauben. Dabei sollte es auch möglich sein, die mobile Middleware über entsprechende Adapter mit klassischen Middleware-Systemen und nicht-mobiler Hardware anbinden zu können (*Kompatibilität*). Dabei sollten Verfahren und Mechanismen zur Energieverwaltung integriert werden, um damit die meist endlichen Energievorräte mobiler Geräte zu schonen (*Energieeffizienz*). Da in einem mobilen System kaum längerfristige Beziehungen zwischen einzelnen Geräten bestehen und ein bedingungsloses Vertrauen unter den Teilnehmern auch nur ein unerreichbares Ideal ist, müssen Mechanismen zum Etablieren von *Vertrauen* ein integraler Bestandteil von Middleware-Systemen im Mobile Computing sein. Aber auch die Kommunikation zwischen zwei sich vertrauenden Partnern bedarf in drahtlos kommunizierenden Systemen zusätzlicher Absicherung, da meist sensible Daten über nicht zu kontrollierende Broadcast-Medien verschickt werden (*Sicherheit*). [Sch05, Sat03, Rot02b, MCE02]

3.2. Asynchrone Kommunikation als Basis mobiler Anwendungen

Die bei der Konzeption vieler traditioneller Middleware-Systeme gemachten Annahmen über eng gekoppelte Kommunikationspartner, die durch permanente und leistungsfähige Netzwerke miteinander verbunden sind, kommen in mobilen Umgebungen jedoch meist nicht vor. Vielmehr ist die physikalische und logische Netzstruktur extrem dynamisch und Kommunikationsverbindungen bestehen meist nur relativ kurze Zeit, sodass die gleichzeitige Erreichbarkeit von Kommunikationspartnern nicht gewährleistet ist. Deshalb muss die Kommunikation in mobilen Verteilten Systemen durch die eingesetzte Middleware möglichst *zeitlich und örtlich entkoppelt* werden. Diese Entkopplung wird meist durch *asynchrone Kommunikationsparadigmen* umgesetzt, die es erlauben, dass Komponenten auch dann miteinander Nachrichten austauschen können, wenn sie nicht gleichzeitig im Netzwerk verfügbar sind. [GB03, MCE02, SAS01, CDN01]

Dabei unterscheiden sich synchrone und asynchrone Kommunikation vor allem in der zeitlichen Kopplung von Sender und Empfänger in einer Kommunikationsbeziehung (vgl. Abbildung 3.3). Während im synchronen Fall der Sender zwischen Versand der Nachricht und dem Eintreffen der Antwort des Empfängers blockiert ist und somit keine aktive Ar-



(a) synchrone Kommunikation (b) asynchrone Kommunikation

Abbildung 3.3.: Interaktionsmuster (nach (Ham05))

beit erledigen kann, bleibt im asynchronen Fall der Sender auch nach dem Versand einer Nachricht aktiv. Durch das Auflösen der Beziehungen zwischen den Nachrichten des Aufrufs und des Ergebnisses erreicht die asynchrone Kommunikation eine weitgehende Entkopplung von Sender und Empfänger und lässt gegenüber der synchronen Kommunikation nur selbstgesteuerte Kommunikationsformen zu. [Ham05]

Im Bereich des Mobile Computing ist eine Vielzahl von verschiedenen kommunikationsorientierten Middleware-Systemen entwickelt worden, die unterschiedliche Methoden zur Entkopplung der Kommunikation einsetzen. Die nächsten Abschnitte führen in verbreitete asynchrone Kommunikationsalternativen ein und stellen jeweils beispielhaft Vertreter des entsprechenden Basiskonzepts vor.

3.2.1. Nachrichtenorientierte Kommunikation

Middleware-Systeme zur Unterstützung nachrichtenorientierter Kommunikation (*Message-Oriented Middleware* / *MOM*) entkoppeln den Informationsaustausch zwischen Kommunikationspartnern durch die Verwendung von Nachrichtenwarteschlangen (*Message Queues*). Zwar sind MOM-Architekturen nicht direkt für mobile Verteilte Systeme entwickelt worden, werden aber aufgrund des inhärenten asynchronen Kommunikationsmodells als geeignete Grundlage für mobile Verteilte Systeme angesehen, da Sender und Empfänger einer Nachricht nicht gleichzeitig an das Kommunikationsnetz angeschlossen sein müssen. [Ham05, KP03, TS02]

Nachrichtenwarteschlangen-Modell Ein Nachrichtenwarteschlangensystem basiert auf der grundlegenden Idee, dass Anwendungen nicht di-

rekt miteinander Informationen austauschen, sondern diese als Nachrichten an eine (ausgehende) Warteschlange innerhalb der (lokalen) Middleware übergeben (vgl. Abbildung 3.4). Diese übermittelt die Nachricht dann, gegebenenfalls über mehrere dazwischen liegende Kommunikationsserver, an die Middleware-Instanz des Empfängers. Dort wird die Nachricht in die lokale (eingehende) Warteschlange eingefügt und kann von den Anwendungen auf der Empfängerseite nach Bedarf aus dieser entnommen werden. Da die Nachrichten meist in allen Warteschlangen, die sie auf ihrem Weg vom Sender zum Empfänger passieren, zwischengespeichert werden, können auch dann Nachrichten ausgetauscht werden, wenn einer der Kommunikationspartner temporär nicht verbunden ist. [TS02, SAS01]

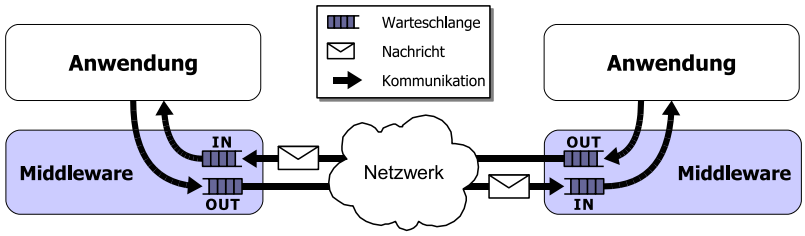


Abbildung 3.4.: Kommunikation mittels Nachrichtenwarteschlangen

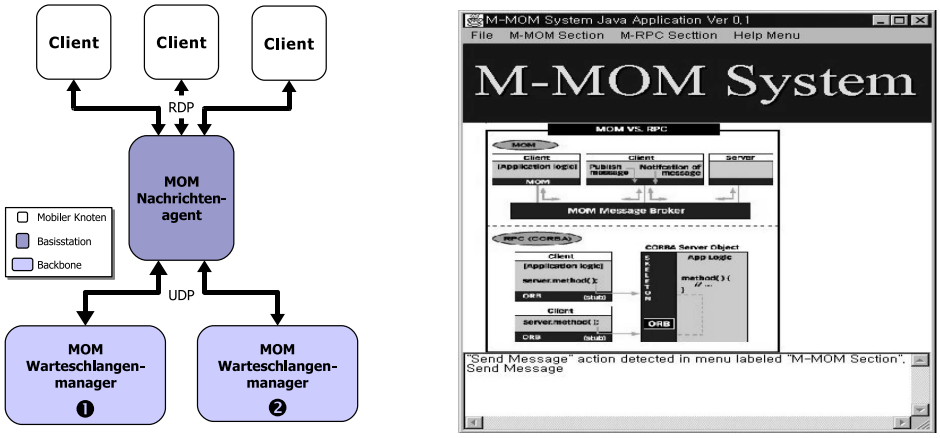
Nachrichten, die über MOM-Systeme versendet werden, können im Prinzip beliebige Inhalte transportieren. Dabei wird zur Adressierung der Zielwarteschlange meist ein transportprotokollunabhängiger und systemübergreifend eindeutiger Name (*Unique Identifier*) verwendet. Jedoch kann die Middleware Auslieferungsgarantien (*Delivery Assurances*) nur bezüglich der Zustellung an die adressierte Warteschlange geben, da die Entnahme der Nachricht und deren Verarbeitung vollständig von der Empfängeranwendung abhängig sind. Die gegebene Garantie legt dabei fest, ob eine Nachricht nur eventuell (*May-Be-Semantik*), höchstens einmal (*At-Most-Once-Semantik*), mindestens einmal (*At-Least-Once-Semantik*) oder genau einmal (*Exactly-Once-Semantik*) zugestellt wird. In reinen Ad-hoc-Systemen (vgl. Abschnitt 2.3.2.1) können im Allgemeinen nur die May-Be- und At-Most-Once-Semantik garantiert werden, da es in einem solchen Netz keine dauerhaften Synchronisationspunkte gibt. Bei Infrastruktursystemen (vgl. Abschnitt 2.3.2.2) hingegen, können aufgrund der statischen Backbone-Komponenten, auch weitergehende Garantien abgegeben werden. [TS02]

Die Reihenfolge für die Entnahme der Nachrichten aus den Warteschlangen kann entweder durch Ordnungsprinzipien, wie beispielsweise das FIFO-Prinzip (*First In, First Out*), durch anwendungsdefinierte Prio-

ritäten oder durch Load-Balancing-Schemata bestimmt sein. Durch die Verwendung von Filtern können Nachrichten auch bereits direkt innerhalb einer Warteschlange herausgefiltert werden, um sie zum Beispiel an andere Server weiterzuleiten oder unerwünschte Nachrichten zu verwerfen, ohne dass die Anwendung die Nachrichten bearbeiten muss. Aber auch Gruppenkommunikationsformen können leicht durch nachrichtenorientierte Middleware-Systeme realisiert werden, indem Nachrichten verdoppelt werden oder Warteschlangen gemeinsam genutzt werden. [JPK99]

3.2.1.1. MOBILE MOM

Die an der *Korea University* entwickelte *MOBILE-MOM*-Plattform (vgl. Abbildung 3.5(b)) stellt eine nachrichtenorientierte Middleware für infrastrukturbasierte mobile Systeme (vgl. Abschnitt 2.3.2.2) dar. Dabei kommunizieren *mobile Clients* dynamisch und asynchron mit *Nachrichtenagenten*, die als Zugangspunkt zu den eigentlichen *Warteschlangenmanagern* im Backbone-Netz dienen (vgl. Abbildung 3.5(a)). Das MOBILE-MOM-System bietet insgesamt eine nicht-blockierende und asymmetrische Kommunikation über persistente Nachrichtenwarteschlangen an. [JPK99]



(a) Systemübersicht

(b) Benutzungsinterface

Abbildung 3.5.: MOBILE MOM (nach/aus (JPK99))

MOBILE MOM ist eine auf IP-basierten Netzen aufbauende und fehlertolerante Middleware, die auf persistenten Nachrichtenwarteschlangen beruht und in Fehlerfällen Nachrichten eigenständig an alternative Warteschlangen weiterleiten kann. Die Anwendungsschnittstelle (API) und die Middleware selbst sind in Java geschrieben und übertragen als Nachrichteninhalte serialisierte Java-Objekte. [JPK99]

Die Systemarchitektur (vgl. Abbildung 3.5(a)) besteht anwendungsseitig aus einer lokalen Stellvertreterkomponente (*Proxy*), die zwei temporäre, aber persistente Warteschlangen enthält, in denen ausgehende und eingehende Nachrichten zwischengespeichert werden, bis das mobile Gerät eine Kommunikationsverbindung zum Middleware-Backend-System aufbauen kann beziehungsweise Nachrichten von der Applikation entnommen werden. Dieser Proxy kommuniziert über das *Reliable Datagram Protokol* (RDP) mit einem *Nachrichtenagenten*, der als Zugangspunkt zum Backend-System in den Basisstationen des Infrastrukturnetzwerkes angesiedelt ist. Der Agentet bietet, neben der Weiterleitung der Nachrichten per *User Datagram Protokol* (UDP) an die Warteschlangen-Manager, einen optionalen Verzeichnisdienst an. Die Warteschlangen-Manager verwalten, je nach Art der angebotenen Dienste, eine oder mehrere persistente Warteschlangen für Java-Objekte. Dabei enthalten die Nachrichten neben den Objekten einen *Object Identifier* (OID) als eindeutiges Identifizierungs- und Adressierungsmerkmal. Der OID ist dabei im MOBILE-MOM-System eindeutig und besteht aus den IP-Adressen und Port-Nummern des Senders und des Warteschlangenmanagers sowie einer monoton wachsenden Sequenznummer, einem Nachrichtentyp und einer Prioritätsangabe. [JPK99]

3.2.1.2. Mobile Message Queue

Um die intrinsischen Probleme des Mobile Computing, wie verhältnismäßig langsame und schlechte Kommunikationsverbindungen (vgl. Abschnitt 2.4), zu kompensieren, hat die Firma *Hitachi* eine Middleware entwickelt, die auf nachrichtenorientierter Kommunikation basiert. Das System ist dabei nach dem Client-Server-Modell entworfen und für den Einsatz in Infrastruktursystemen entwickelt worden (vgl. Abbildung 3.6). Es dient dabei unter anderem zum mobilen Datenbankzugriff. [MTK99, SMNT99]

Die eigentliche Verwaltung der Warteschlangen wird von der *Hitachi Nachrichtenwarteschlangen-Software für OLTP* (Online Transaction Processing) realisiert. Um diese auch mobilen Anwendungen zugänglich zu machen, ist das System um Komponenten zum mobilen Zugriff auf Nachrichtenwarteschlangen-Systeme erweitert (*Mobile Message Queue Software*). Dabei gibt es jeweils eine angepasste Komponente für die mobilen Clients und die stationären Server. Diese als Agenten bezeichneten lokalen Komponenten ermöglichen den asynchronen Austausch von Nachrichten zwischen mobilen dem Client und dem Server. Dabei wird der relativ unsichere drahtlose Informationsaustausch durch die Verwen-

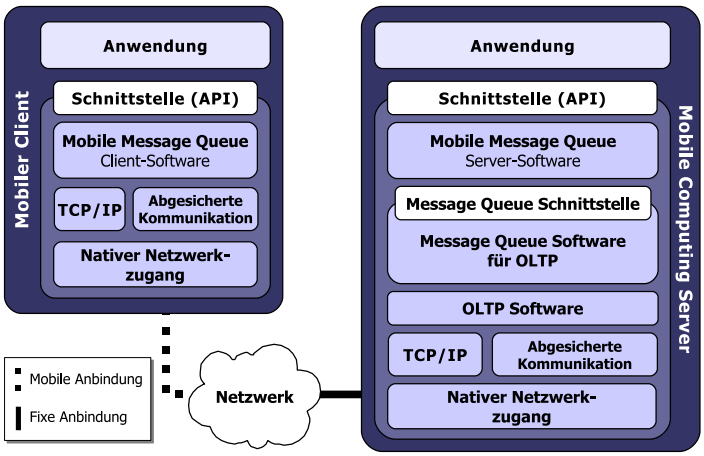


Abbildung 3.6.: Architektur der Mobile Message Queue (nach (MTK99, SMNT99))

dung von *abgesicherten Kommunikationsverbindungen* geschützt. Diese Absicherung beinhaltet dabei die Authentifizierung der Kommunikationspartner, die Verschlüsselung der Kommunikation sowie die Komprimierung aller zu übertragenden Daten. Der Zugriff der Anwendungen auf die Mobile-Message-Queue-Agenten wird durch eine einheitliche Programmierschnittstelle (API) realisiert. [MTK99, SMNT99]

3.2.2. Ereignisbasierte Kommunikation

Systeme mit ereignisbasierter Kommunikation, die auch als *Publish/Subscribe-Systeme* bezeichnet werden, verbinden autonome Kommunikationspartner durch den Austausch von Ereignisnachrichten. Dabei übermitteln Komponenten, die als *Nachrichtenquelle* agieren, Informationen über in ihrem Ausführungskontext wahrgenommene Zustandsänderungen in Form von Ereignismeldungen (*Events*) an einen oder mehrere interessierte Abnehmer. Die Vermittlung zwischen der Ereignisquelle und dem Abnehmer übernimmt dabei ein dezidierter Vermittlungsdienst, an dem der Abnehmer sein Interesse an einzelnen Ereignissen oder Ereignisklassen bekunden kann. Eine solche ereignisbasierte Kommunikation ist vor allem durch ihre Asynchronität, eine lose Kopplung von anonymen Kommunikationspartnern und ihre inhärente Multicast-Fähigkeit charakterisiert. Anonymität herrscht deshalb zwischen den Komponenten, weil weder der Ereignisproduzent einen speziellen Abnehmer adressieren noch der Abnehmer die Quelle des Ereignisses kennen muss. [HGM04, CDN01]

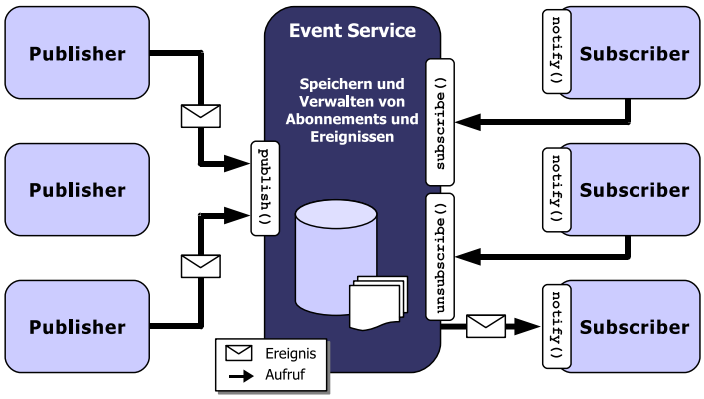


Abbildung 3.7.: Grundprinzip eines ereignisbasierten Systems (nach (EFGK03))

Publish/Subscribe-Modell Das ereignisbasierte Kommunikationsmodell (vgl. Abbildung 3.7) ermöglicht es Abonnenten (*Subscriber*), ihr Interesse an einzelnen Ereignisarten oder einer Klasse von Ereignissen auszudrücken, um dann über das nachfolgende Auftreten der spezifizierten konkreten Ereignisse unterrichtet zu werden. Dazu registriert (`subscribe()`) der Subscriber sein Interesse bei einer neutralen Vermittlungskomponente, dem so genannten *Event Service*, der die Abonnements aller Subscriber speichert und verwaltet. Tritt bei einer Quellkomponente (*Publisher*) ein Ereignis auf, so veröffentlicht sie dieses (`publish()`) ebenfalls mit Hilfe des Event Services. Dieser hat dann die Aufgabe, die an ihn übermittelte Ereignisnachricht an alle bei ihm für dieses Ereignis registrierten Subscriber weiterzuleiten (`notify()`). Der Event Service stellt somit eine Stellvertreterkomponente (*Proxy*) für alle Abonnenten dar und entkoppelt den Publisher von den Subscribern sowohl örtlich, da sie keine Referenzen aufeinander haben, als auch zeitlich, da sie nicht gleichzeitig mit dem Event Service verbunden sein müssen. Dies führt dazu, dass Publish/Subscribe-Systeme besonders gut skalieren und sich besonders gut an Veränderungen anpassen können. [EFGK03, HGM04, CDN01]

Da Abonnenten meist nicht an allen Ereignisnachrichten, die über den Event Service vermittelt werden, interessiert sind, müssen sie die Möglichkeit besitzen, die gewünschte Menge zu spezifizieren. Dazu haben sich verschiedene Abonnementsschemata herausgebildet, wie *themenbasiertes*, *inhaltsbasiertes* oder *typbasiertes Abonnieren*. Beim themenbasierten Abonnieren werden Ereignisse und Abonnements einzelnen Themenbereichen durch die Angabe eines Schlagwortes zugeordnet. Diese Art der Zuordnung führt zu einer Gruppenkommunikation, in welcher der Adressatenkreis relativ statisch festgelegt ist, da immer alle Gruppenmitglieder dieselbe Er-

eignisnachricht erhalten. Bei der inhaltsbasierten Variante wird hingegen nicht ein festes äußeres Kriterium ausgewählt, sondern über die Eigenschaften des Ereignisses selbst der Empfängerkreis bestimmt. Je nachdem wer an welcher Eigenschaft des Ereignisses interessiert ist, verändert sich die Menge der Empfänger dynamisch. Eine solche dynamische Zuordnung der Ereignisse geschieht auch beim typbasierten Abonnieren, das sich an dem aus Programmiersprachen bekannten Typkonzept orientiert. Damit lassen sich durch den Typ bestimmte Ereignisse sehr leicht in Programmiersprachen integrieren und Typeigenschaften wie die Vererbung zur Auswahl von relevanten Ereignissen nutzen. [HGM04]

3.2.2.1. Das REBECA-System

Die *REBECA*-Plattform der *Technischen Universität Darmstadt* ist ein an die Bedürfnisse des Mobile Computings angepasstes Publish/Subscribe-System mit inhaltsbasierten Ereignisabonnements. Dabei stützt es sich auf ein infrastrukturbasiertes *verteiltes Event-System*, das die physikalische Mobilität von Clients durch *Handover-Mechanismen* unterstützt und zudem *ortbasierte Filtermechanismen* anbietet. [FGKZ03, ZF03]

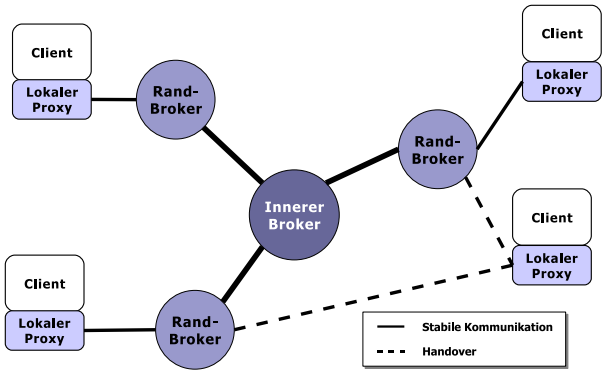


Abbildung 3.8.: Aufbau des REBECA-Vermittlungssystems (nach (AAGC04))

Zugang zur nachrichtenorientierten Kommunikationsplattform erhalten (mobile) Clients durch einen lokalen Event Broker, der als *Proxy* auf Seiten der Teilnehmer dient (vgl. Abbildung 3.8). Dieser bietet eine einfache Programmierschnittstelle, über welche die Grundfunktionen des Abonnierens, des Kündigens von Abonnements und des Veröffentlichens von Ereignissen realisiert werden. Benachrichtigungen über das Eintreten von Ereignissen werden mittels einer Call-Back-Methode realisiert. Diese lokalen Proxies sind mit dem aus einem *Netz aus Brokern* bestehenden Event System verbunden und können so miteinander Ereignisnachrichten austau-

schen. Die Broker des Event-Systems sind (logisch) in einem azyklischen und zusammenhängenden Graphen organisiert. Dabei unterteilen sich die Knoten des Graphen in *Rand-Broker*, die Verbindungen zu Clients besitzen, und in *innere Broker*, die lediglich mit anderen Brokern verbunden sind. [FGKZ03, ZF03]

Abonnenten beschreiben im REBECA-System ihr Interesse an Ereignissen durch das Registrieren von *Filtern*, die boolesche Funktionen über sämtliche Inhalte einer Ereignisnachricht darstellen. Dabei abonnieren die Clients Ereignisnachrichten direkt an ihrem Randknoten, der dann wiederum den im Abonnement übergebenen Filter nutzen kann, um sein Interesse an Ereignisnachrichten an benachbarte Broker weiterzugeben. Je nachdem wie die Filter der Subscriber vom Broker genutzt werden, um eigene Filter zu generieren, unterscheidet man verschiedene Strategien zur Wegewahl der Ereignisse im REBECA-Event-System: Beim *Simple Routing* werden alle neuen und aufgehobenen Abonnements durch das gesamte Broker-Netz „geflutet“. Durch das *Identity Routing* werden identische Abonnements auch bei öfterem Auftreten nur einmal weitergeleitet. Die Strategie des *Covering Routing* verhindert die Weitergabe von Abonnements, deren Filter bereits von vorhergehenden Abonnements abgedeckt sind. Besonders komplex ist die *Merge-Routing*-Strategie, da diese aus den Filtern aller Subscriber eines Brokers einen kombinierten Filter generiert, der dann vom Broker selbst bei seinen Nachbarn als Abonnement registriert wird. [AAGC04, FGKZ03]

Die physikalische Mobilität der Clients wird durch eine in das REBECA-System integrierte *Handover-Methode* unterstützt. Dabei ist der Algorithmus so gewählt, dass Clients ihre Abonnements beim Wechsel des Rand-Brokers nicht erneuern müssen. Vielmehr aktualisiert sich die Konfiguration des Broker-Netzes automatisch mit der neuen Zugangsposition des Clients. Dabei werden Ereignisse, die während der Handover-Zeit eintreten, vom ursprünglichen Rand-Broker des Clients gesammelt und dann über den neuen Randknoten an den Client ausgeliefert. [ZF03]

3.2.3. Kommunikation über assoziative und geteilte Speicher

Die Verwendung von geteilten und assoziativ adressierbaren Speichern – zum Beispiel in Form eines *Tuple Space* oder *Blackboards* – bietet eine weitere Möglichkeit, die synchronen Kommunikationsparadigmen traditioneller Verteilter Systeme durch asynchrone zu ersetzen. Durch die sowohl in der Zeit als auch im Ort entkoppelte Kommunikation werden Tuple-

Space-basierte Systeme auch im Bereich des Mobile Computing eingesetzt. [Cap02, MCE02, EM88]

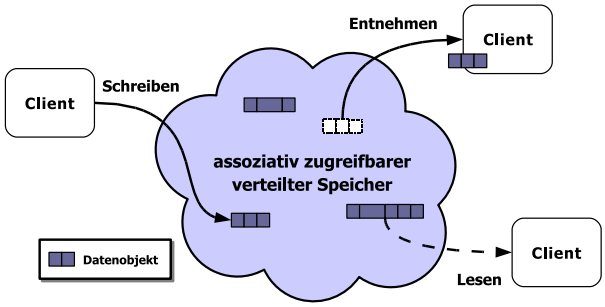


Abbildung 3.9.: Arbeitsweise eines Tuple Spaces (nach (FHA99, FWDB97))

Das Konzept, einen globalen, gemeinsam genutzten und assoziativen Speicher zur Kommunikation zu verwenden, basiert auf dem *Linda-Programmiermodell* zur Koordination paralleler Prozesse der *Universität Yale*. Dieser verteilte Speicher wird dabei zum Austausch von Datentupeln zwischen lokalen und entfernten Prozessen verwendet (vgl. Abbildung 3.9). Ein Tupel repräsentiert hierbei einen anonymen Vektor typisierter Werte. Das Tupel unterliegt nach der Übergabe an den gemeinsamen Tuple Space einem eigenen Lebenszyklus, der von dem Prozess unabhängig ist, der das Tupel zuvor erzeugt hat. Zudem wird es persistent gespeichert, bis ein beliebiger Prozess das Tupel wieder aus dem gemeinsamen Speicher entnimmt, wobei es zwischenzeitlich beliebig oft gelesen werden kann. Diese Persistenz der Tupel führt dazu, dass Kommunikationspartner nicht gleichzeitig am Tuple Space teilhaben müssen und somit ihr Datenaustausch zeitlich entkoppelt wird. [WCC04, MCE02, Win01, FWDB97]

Der Zugriff auf die Daten im Tuple Space erfolgt, da die Einträge keine festgelegte Identität besitzen, assoziativ über Schablonen (*Antitupel*). Alle sich im verteilten Speicher befindlichen Tupel, die einer solchen Schablone genügen, stellen das Ergebnis einer entsprechenden Anfrage dar. Somit ist eine Kommunikation ohne explizite Adressierung und sogar ohne explizite Kenntnis der Kommunikationspartner untereinander möglich. Sollte dabei eine Schablone auf mehrere Tupel passen, wird eines hiervon nicht-deterministisch als Ergebnis der Anfrage gewählt. [WCC04, EFGK03, FWDB97]

3.2.3.1. LIME – Linda in a Mobile Environment

Das an der *Universität von Washington* entwickelte LIME-*System* stellt eine Implementierung des Tupel-Space-Ansatzes für mobile Einheiten dar.

LIME rückt dabei von dem Ansatz eines einheitlichen und globalen geteilten Speichers zugunsten *lokaler und transient föderierter Tuple Spaces* ab, um die intrinsischen Eigenschaften mobiler Systeme (vgl. Abschnitt 2.4) besser zu unterstützen. Insbesondere kann jeder Nutzer des Systems mehrere benannte Tuple Spaces mit unterschiedlichen Verteilungsgraden besitzen. Dabei ist der LIME-Ansatz so aufgebaut, dass nicht zwischen der physikalischen Mobilität eines Endgeräts und der logischen Mobilität von Code-Fragmenten unterschieden werden muss. [MCE02, PMR99]

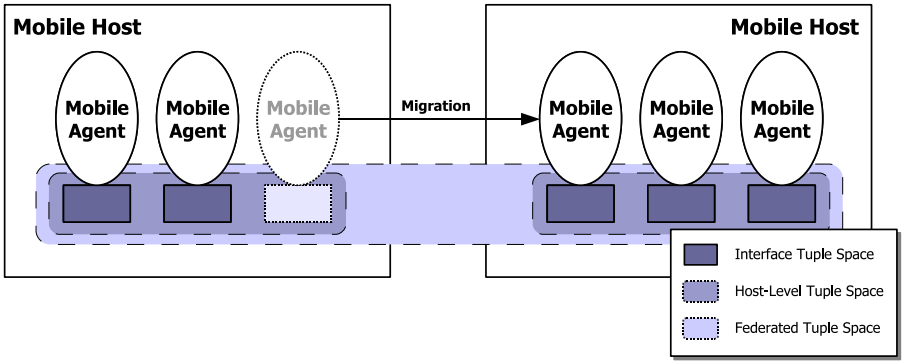


Abbildung 3.10.: LIME-Systemübersicht (nach (PMR99))

Jeder Klient des Systems greift auf seine lokalen Daten über Schnittstellen-Tuple-Spaces (*Interface Tuple Spaces*) zu, die ihm auch beim möglichen Wechsel des Endgerätes stets zugeordnet bleiben (vgl. Abbildung 3.10). Die Daten innerhalb eines solchen lokalen Tuple Space können dabei vom Klienten – je nach Bedarf – als öffentlich oder privat gekennzeichnet werden, um Daten mit anderen auszutauschen beziehungsweise den Austausch zu verhindern. Die Daten des lokalen Tuple Space werden vom LIME-System transient mit den Daten anderer Klienten erweitert, indem passende Tuple Spaces aller Klienten eines Endgerätes zu einem (virtuellen) Tuple Space auf Endgeräteebene (*Host-Level Tuple Space*) vereinigt werden. In gleicher Weise werden auch diese Tuple Spaces auf Endgeräteebene von sich in Kommunikationsreichweite befindenden Geräte zu einem föderierten Tuple Space (*Federated Tuple Space*) vereinigt und so der Datenaustausch zwischen Klienten auf unterschiedlichen Geräten ermöglicht. Die durch logische oder physikalische Mobilität hervorgerufene Vereinigung verschiedener Tuple Spaces (*Engagement*) beziehungsweise das Herauslösen eines nicht mehr erreichbaren Tuple-Space (*Disengagement*) führt zu der transienten Natur der kombinierten Tuple Spaces auf höherer Ebene, aber auch zur Aufgabe der Persistenz nicht lokaler Tupel. [MPR06, PMR99]

Durch das Einbinden von zusätzlichen Tupelmetainformationen, wie

zum Beispiel der Lokation eines Tupels, und dem Zugriff auf Systemkonfigurationen mittels eines nur lesbaren System-Tuple-Spaces wird in LIME versucht, einen erhöhten Grad an Kontextbewusstsein in das Tuple-Space-Modell zu integrieren. Zusätzlich bietet LIME die Möglichkeit, Code-Fragmente proaktiv vom LIME-System ausführen zu lassen, falls eine Entsprechung für ein gegebenes Antitupel in einem Tuple Space erscheint. Diese *reaktive Programmierung* ermöglicht es Klienten – zumindest ansatzweise – eine ereignisbasierte Kommunikation (vgl. Abschnitt 3.2.2) zu etablieren. [MPR06, PMR99]

3.3. Dynamische Adaption mobiler Systeme durch Kontextbewusstsein

Die hohe Dynamik mobiler Ausführungsumgebungen hat zu der Erkenntnis geführt, dass Anwendungen in solchen Systemen sich zunehmend ihrer Mobilität bewusst werden und sich den stattfindenden Veränderungen anpassen sollten (vgl. Abschnitt 2.5.2). Um ein solches Bewusstsein zu erlangen, nutzen mobile Anwendungen und Systeme Informationen aus ihrem so genannten (Ausführungs-)Kontext, der abstrakt folgendermaßen definiert werden kann:

> ***Context*** *is any information that can be used to characterize the situation of an entity. An entity is a person, place, or physical or computational object.*
>
> [Dey00, Seite 79]

Damit bezeichnet der Kontext jegliche Information, die einen Zustand beschreibt, der für eine Anwendung oder den Nutzer relevant ist. Dabei umfassen diese Informationen neben Daten über die vorhandene Infrastruktur (*Computing Context*) und den Nutzer (*User Context*) auch Daten über die physikalische Umgebung (*Physical Context*), wie die Helligkeit oder die Temperatur. In diesem Zusammenhang können Entitäten je nach Kontext unterschiedlich interpretiert werden und insbesondere auch Teil ihres eigenen Kontextes sein. [CK00, DA99]

Die Daten eines Kontextes können grob in primäre und sekundäre Informationen unterteilt werden. Primäre Daten – wie der Ort, die Identität oder die Zeit – dienen einzeln oder kombiniert als eine Art Schlüssel beziehungsweise Index, um auf Entitäten des Kontextes zuzugreifen. Weitere Eigenschaften der Entitäten werden dann als sekundäre Kontextdaten bezeichnet, auch wenn sie eventuell eine größere Relevanz für ei-

ne spezielle Anwendung haben. Orthogonal hierzu werden Kontextinformationen auch in ihrem Abstraktionsgrad unterschieden: *Low-Level-Kontextdaten* bezeichnen dabei direkt durch logische oder physikalische Sensoren messbare Daten wie die Zeit, die verfügbare Bandbreite eines Übertragungskanals oder die Helligkeit. *High-Level-Kontextdaten* hingegen sind von Low-Level-Kontextdaten abgeleitet und geben komplexere (soziale) Zustände wieder. So kann zum Beispiel durch die aktuelle Position eines Nutzers zusammen mit der Zeit und Einträgen des persönlichen Kalenders darauf geschlossen werden, dass der Nutzer gerade an einer Konferenz teilnimmt. Das Erschließen solcher Kontextdaten auf höherer Ebene ist jedoch meist nur in abgegrenzten Domänen sinnvoll, da eine allgemeine Interpretation durch Mehrdeutigkeiten oftmals sehr erschwert wird und so die Fehlerrate eine unakzeptable Höhe annehmen kann. [BN04, CK00, DA99, SAT+99, SBG98]

Nutzen Anwendungen während ihrer Ausführung Informationen über relevante Entitäten ihres Kontexts werden sie als *kontextbezogen* bezeichnet:

> *An application is **context-aware** if it adapts its behavior depending on the context.*
>
> [Bec04, Seite 13]

Kontextbezogene Anwendungen können für sich relevante Kontextinformationen entweder aktiv oder passiv nutzen. Ein *aktives Kontextbewusstsein* liegt dann vor, wenn eine Anwendung ihr Verhalten bewusst an einen geänderten Kontext anpasst. Wird eine solche Änderung jedoch nur angezeigt oder persistent gemacht, liegt hingegen ein *passives Kontextbewusstsein* der Anwendung vor. [CK00]

3.3.1. Klassifizierung kontextbezogener Systeme

Kontextbezogene mobile Systeme können Informationen des Kontextes auf unterschiedliche Weise nutzen, um ihr Verhalten ihrer Umgebungssituation anzupassen. Allgemein haben sich dabei folgende Klassen der Kontextnutzung etabliert:

Kontextbezogene Selektion Systeme und Anwendungen dieser Klasse nutzen Kontextwissen bei der Auswahl von Informationen und Diensten. Im Allgemeinen besitzen dabei Angaben zum Ort beziehungsweise zum Abstand zu anderen Systemen, die Elemente in der unmittelbaren Umgebung und persönliche Präferenzen eine besonders hohe Relevanz. Anwendungen dieser Klasse kontextbezogener Systeme sind zum Beispiel

elektronische Touristen- oder Museumsführer, wie in den Projekten *GUIDE* [CDMF00] und *Cyberguide* [AAH+97], oder auch Systeme, die in der Nähe gelegene Peripheriegeräte (beispielsweise einen Drucker) nutzen. [KZL07b, Bec04, RBB03, DA99]

Kontextbezogene Präsentation Verändern Systeme die Art, wie sie Informationen präsentieren, entsprechend ihres Kontextwissens, so spricht man von kontextbezogener Präsentation. Dabei wird meist die Art der Darstellung oder der Detailgrad der aktuellen Situation angepasst. Navigationsanwendungen dieser Klasse können beispielsweise die Ausführlichkeit ihrer Wegbeschreibung der Geschwindigkeit des Nutzers anpassen (*REAL-Projekt* [BKW02]). Ein weiteres Beispiel sind Mobiltelefone, die während einer Besprechung automatisch von akustischer zu optischer Anzeige ankommender Anrufe umschalten (*Context-Call* [STM00]). [KZL07b, Bec04, RBB03, DA99]

Kontextbezogene Aktion Systeme dieser Klasse passen den von ihnen bereitgestellten Dienst automatisch an Veränderungen des Kontexts an (im Besonderen Veränderungen der Güte verwendeter Dienste (QoS)). So wird zum Beispiel im Projekt *UbiQoS* [BST04] die Bildrate eines Video-Streams automatisch an die verfügbare Bandbreite des Übertragungskanals angepasst. Aber auch Anwendungen im Bereich intelligenter Gebäude (*Smart Homes and Buildings*) [SSJS03], die ihre Konfiguration (Beleuchtung, Temperatur etc.) automatisch den Präferenzen der anwesenden Personen anpassen, fallen in diese Klasse. [KZL07b, Bec04, RBB03, DA99]

Kontextbezogene Annotation Anwendungen und Systeme, die diese Klasse bilden, besitzen die Fähigkeit, Kontexte, im Besonderen enthaltene physikalische Entitäten oder spezielle Situationen, mit digitalen Daten oder Informationen anzureichern beziehungsweise zu assoziieren. Diese Daten werden dann, wenn sich der assoziierte Kontext abermals einstellt, dem Benutzer erneut angezeigt. Dieses Prinzip der virtuellen Merkzettel ist zum Beispiel im Projekt *Stick-e Notes* [Pas97] umgesetzt. [KZL07b, Bec04, DA99]

Im Allgemeinen ist das Kontextbewusstsein und die Nutzung dieser Informationen zur (automatischen) Anpassung von Anwendungen momentan auf die Unterstützung mehr oder weniger monolithischer und ad-hoc-statischer[1] Anwendungen bei der Ausführung kurzzeitiger Aufgaben be-

[1] Eine Anwendung beziehungsweise ein Prozess ist *ad-hoc-statisch*, wenn seine Ziele und Teilnehmer zum Startzeitpunkt dynamisch ermittelt werden, danach jedoch alle Akti-

schränkt. Dies bedeutet, dass die Informationen über verfügbare Ressourcen meist nur dazu verwendet werden, um eine Anwendung einmalig mit geeigneten Diensten oder Informationen zu parametrisieren und so eine dem Kontext angepasste Ausführung zu erhalten. [KZL07b]

3.3.2. Kontextmodelle

Ein Schlüsselfaktor eines jeden kontextbasierten Systems ist das ihm zugrunde liegende Kontextmodell. Es dient dazu, Anwendungen vom Prozess der Kontextdatengewinnung und -verarbeitung zu separieren und ermöglicht so die Nutzung der Kontextdaten auch über verschiedene Anwendungsgrenzen hinweg. Dabei werden die Informationen nicht nur aus physikalischen oder logischen Sensoren gewonnen, sondern werden auch aus den Anwendungen selbst bezogen (vgl. Abbildung 3.11). Das Modell legt hierbei fest, wie die im Kontext enthaltenen Informationen beschrieben und gegebenenfalls zu höheren Kontextinformationen aggregiert oder abgeleitet werden können. Zusammen mit einem entsprechenden Kontext-Management-System kann so eine universelle Plattform zur geräteübergreifenden Nutzung von Kontexten entstehen. [SLP04, LBBN04, BN04, RBB03]

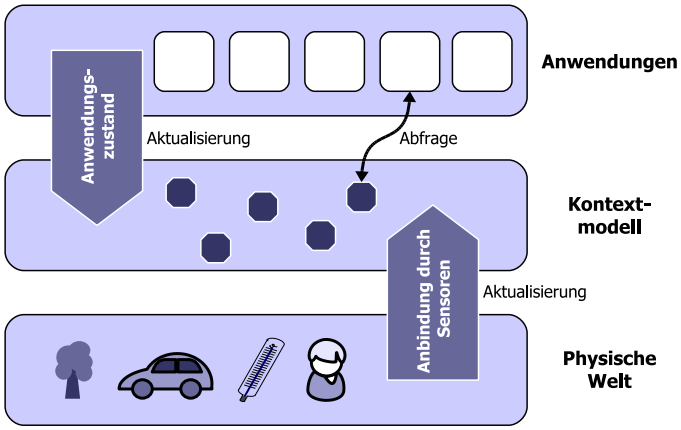


Abbildung 3.11.: Kontextmodellierung (nach (BN04))

Der Zugriff auf die im Kontextmodell enthaltenen Daten kann entweder durch gezielte Anfragen (*Queries*) oder durch asynchrone Kommunikationsparadigmen, wie zum Beispiel dem *Publish / Subscribe-Paradigma* (vgl. Abschnitt 3.2.2), erfolgen. [BN04]

Aus Anwendungs- beziehungsweise Nutzersicht lassen sich Kontextmodelle anhand ihrer *räumlichen Ausdehnung*, der *Komplexität der Modellab-*

vitäten statisch abgearbeitet werden. [Rie05]

straktion und ihrer *Dynamik* klassifizieren. Dabei gibt die Ausdehnung den geografischen Bereich an, der vom Kontextmodell erfasst wird. Die Komplexität des Modells bezieht sich hingegen auf den unterstützten Detailgrad der abbildbaren Daten und die Dynamik auf die Rate, mit der sich die verfügbaren Kontextinformationen im Modell ändern. Hierzu orthogonal können die Modelle auch aus softwaretechnischer Sicht nach ihren grundlegenden Datenstrukturen gruppiert werden. Dabei haben sich einfache Modelle, wie das *Key-Value-Modell* bis hin zu komplexeren Modellen auf der Basis von Auszeichnungssprachen (*Markup Languages*) oder *Ontologien* sowie *objektorientierte und logikbasierte Ansätze* entwickelt. Gemein ist den Modellen jedoch, dass sie aufgrund der zugrunde liegenden hohen Dynamik der Umgebung besonderen Anforderungen unterliegen und der Kontext demzufolge nur als partiell valid, nicht umfassend und relativ vage angesehen werden muss. Zudem variieren die von Sensoren gelieferten Daten oftmals in ihrer Verfügbarkeit und Qualität. [SLP04, BN04, RBB03]

3.3.2.1. Anwendungsabhängige Modelle

Eine Vielzahl von momentan verfügbaren Infrastrukturen zur Unterstützung des Entwicklungsprozesses kontextbasierter Systeme sind speziell auf eine Anwendungsdomäne bezogen. Anwendungen, die auf solchen Infrastrukturen aufbauen, besitzen entsprechend meist ein nur für ihre Aufgabe entwickeltes und damit geschlossenes Kontextmodell. Eine Kooperation zwischen unterschiedlichen Anwendungen oder die gemeinsame Nutzung von Kontextressourcen ist bei solchen Modellen meist nicht möglich, da sie auf unterschiedlichen Metamodellen[2] und damit auf inkompatiblen Kontextstrukturen basieren. [FHKB05, SLP04, LBBN04]

3.3.2.2. Abstrakte und universell verwendbare Modelle

Mit zunehmender Verbreitung von kontextbezogenen Systemen ist auch der Bedarf gestiegen, die geschlossenen Strukturen von anwendungsabhängigen Kontextsystemen aufzubrechen und eine einheitliche und unabhängige Repräsentierung des Kontexts zu etablieren. Denn nur mit einem einheitlichen Kontextmodell und -verständnis können Kontextdaten zwischen unterschiedlichen kontextbezogenen Anwendungen eines Gerätes ausgetauscht werden. [FHKB05]

Um dies zu erreichen wird das Kontextmodell und die zugehörige Verwaltungskomponente meist von den Anwendungen getrennt und als ei-

[2] Metamodelle sind Modelle, die die Struktur anderer Modelle beschreiben. [Mel04]

genständiger Dienst realisiert, sodass – zumindest logisch – eine Schichtenarchitektur entsprechend der Abbildung 3.11 entsteht. Der Zugriff aus den Anwendungen heraus erfolgt dann über eine einheitliche Schnittstelle, mit der Kontextinhalte abgerufen, aktualisiert und verwaltet werden können.

Dabei wird die Interaktion zwischen unabhängigen Anwendungen dadurch erschwert, dass jede Anwendung meist eine eigene subjektive Sicht auf die Umgebung hat. Dies bedeutet, dass einzelne Anwendungen unterschiedliche Kontextmerkmale als relevant erachten, dass verschiedene Anwendungen unterschiedliche Werte an ein Kontextattribut binden möchten oder aber, dass sie semantisch unterschiedliche Interpretationen für ein Kontextdatum haben. Ein allgemeines und generisches Kontextsystem muss es also ermöglichen, dass auf Basis eines gemeinsamen Kontextmodells jede Anwendung ihre eigene subjektive Sicht auf die im Modell gespeicherten Daten erhält und gegebenenfalls zwischen diesen Sichten vermitteln. [BBB01]

3.3.2.3. Föderierte Kontextmodelle

Durch die Verwendung von abstrakten und generischen Kontextmodellen erhält man eine einheitliche Datenbasis für unterschiedliche Anwendungen eines Geräts, welche jedoch durch die dem einzelnen Gerät zur Verfügung stehenden Datenquellen begrenzt ist. Um diese zunächst isolierten Daten in einem globalen Modell zu vereinen und so den Anwendungen eine globale Sicht auf die Kontextinformationen zu ermöglichen, müssen die lokalen, oft jedoch heterogenen Kontextmodelle zu einem Gesamtmodell vereinigt werden. Bei einer solchen Föderation von Kontextmodellen können sich die Modelle nicht nur ergänzen, sondern auch durch eine (geografische) Überlappung Inkonsistenzen erzeugen. Sie bietet aber auch die Möglichkeit, unterschiedliche Informationsräume, wie zum Beispiel das Internet, zu integrieren. [LBBN04, RBB03]

Die Föderation einzelner Kontexte zu einem potenziellen '*globalen digitalen Weltmodell*' stellt dabei hohe Anforderungen an ein vereinigtes Kontextmodell und -verwaltungssystem. Dabei müssen vor allem semantische Heterogenitäten, Konsistenzproblematiken und zeitliche Aspekte berücksichtigt werden. [RBB03]

3.3.3. Middleware-Systeme zur Unterstützung der Nutzung von Kontextinformationen

An dieser Stelle eine vollständige Übersicht über alle kontextbasierten Systeme zu geben, ist aufgrund ihrer großen Anzahl nicht möglich. Deshalb werden im Folgenden Systeme vorgestellt, die sich dadurch auszeichnen, dass sie ein anwendungsunabhängiges Kontextmodell bereitstellen, wodurch sie als Basis möglichst vieler und unterschiedlicher Applikationen dienen können. Die Plattformen sind so gewählt, dass möglichst unterschiedliche Kontextmodellierungs- und Management-Paradigmen präsentiert werden.

3.3.3.1. Context Toolkit

Das am *Georgia Institute of Technology* entwickelte *Context Toolkit* ist ein Rahmenwerk, mit dem kontextbezogene Anwendungen auf abstrakte Weise und somit generisch entwickelt werden können, ohne sich direkt mit der Erhebung von Kontextdaten befassen zu müssen. Dazu stellt das Context Toolkit eine Kombination von abstrakten Konzepten und konkreten Komponenten bereit, um die Entwicklung von kontextbezogenen Anwendungen über den gesamten Entwicklungszyklus zu unterstützen. Auch hier wird konzeptionell das Kontextmodell und die Verwaltung der Kontextdaten von den einzelnen Anwendungen gelöst und als eigenständige Dienstkomponente angesehen (vgl. Abschnitt 3.3.2). Der Kontextdienst abstrahiert dabei von der physikalischen Erhebung der Kontextdaten, stellt diese Informationen über generische Mechanismen zur Verfügung, gewährt dabei einen transparenten Zugriff auf verteilte Kontextquellen und bietet die Möglichkeit, die Heterogenität der Datenrepräsentationen durch zusätzliche Abstraktionen zu überwinden. [Dey01, Dey00]

Die grundlegende Architektur des Context Toolkit besteht aus drei Komponententypen, die unterschiedliche Elemente eines Kontextes repräsentieren und jeweils autonom agieren (vgl. Abbildung 3.12): *Context Widgets* kapseln als elementare Komponente der Architektur eine einzelne physikalisch erfassbare Kontextinformation und somit Low-Level-Kontextdaten. Sie bieten nach außen eine uniforme Schnittstelle, mit der auf die enthaltenen Daten zugegriffen werden kann. Die Widgets stellen neben dem aktuellen Wert der von ihnen repräsentierten physikalischen Information auch historische Werte bereit, die andere Komponenten der Architektur oder kontextbezogene Anwendungen weiter verarbeiten können. So genannte Verdichterkomponenten (*Aggregators*) führen Kon-

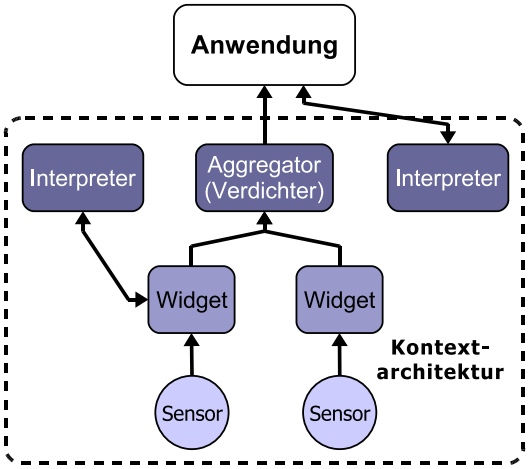


Abbildung 3.12.: Architektur des Context Toolkit (nach (DSFA99))

textinformationen, die zu einer Entität gehören und aus unterschiedlichen Widgets stammen, zusammen und dienen so als Stellvertreterobjekt (Proxy) für die Entität im Kontextmodell. Der dritte Komponententyp im Context Toolkit sind *Interpreter*, die an sie übergebene Kontextinformationen verarbeiten und dabei in andere Formate oder Bedeutungen transformieren (zum Beispiel die Umrechnung von Grad Celsius in Grad Fahrenheit). Hierbei kann zum einen lediglich eine Abbildung zwischen unterschiedlichen Repräsentationen einer Kontextinformation stattfinden. Zum anderen können aber auch High-Level-Kontextdaten aus mehreren anderen Kontextdaten abgeleitet werden. Interpreter können dabei sowohl von Anwendungen als auch von Widgets selbst aufgerufen werden. Zudem wird an der Integration einer weiteren Abstraktion gearbeitet, die eine spezielle Konstellation von Widget- und Verdichterzuständen darstellt und eine *Situation* beschreibt. [Dey01, Dey00, DSFA99]

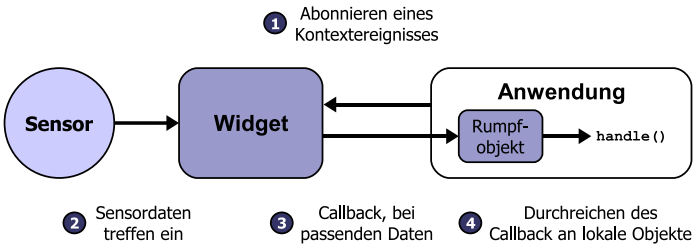


Abbildung 3.13.: Interaktion im Context Toolkit (nach (DSFA99))

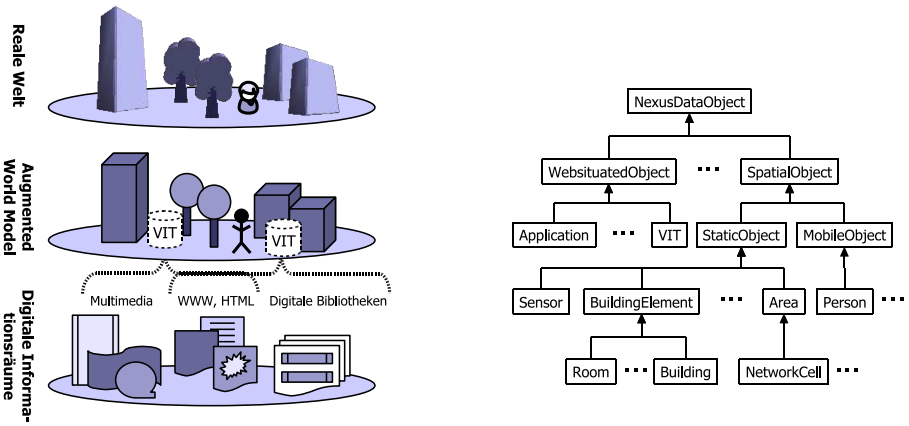
Die Interaktion zwischen Verdichtern und Widgets auf der einen sowie kontextbezogenen Anwendungen auf der anderen Seite geschieht entweder mittels expliziter Anfragen (*Poll-Methode*) oder durch asynchrone Ereignis-

nachrichten (*Subscription-Methode*). Dabei kann eine Anwendung bei der Registrierung ihres Benachrichtigungswunsches zusätzliche Bedingungen (*Conditions*) angeben und so die eingehenden Nachrichten auf die von ihr benötigten Einzelereignisse beschränken (vgl. Abbildung 3.13 (1)). Eingehende Sensordaten (2) werden vom Widget, wenn sie den Bedingungen entsprechen, an ein sich auf Anwendungsseite befindendes Rumpfobjekt in Form einer Ereignisnachricht weitergereicht (3). Dieses Stellvertreterobjekt ruft dann die eigentliche Call-Back-Methode der Anwendung auf (4), sodass die Anwendung auf die Kontextänderung reagieren kann. Obwohl die Referenzimplementierung des Context Toolkit in der Programmiersprache Java geschrieben ist, findet jegliche Kommunikation zwischen den Komponenten sprachunabhängig statt, sodass die Interaktion zwischen Implementierungen in anderen Programmiersprachen möglich ist. [Dey01, DSFA99]

3.3.3.2. Nexus

Die *Nexus-Plattform* hat zum Ziel, eine Vielzahl an kontextbasierten – vor allem aber ortsbasierten – Anwendungen durch ein gemeinsam genutztes globales Kontextmodell, das allgemeine und anwendungsabhängige Kontextdaten vereinigt, zu unterstützen. Dabei ist das System so konzipiert, dass lokale und somit partielle Kontextmodelle einzelner Kontextanbieter (*Context Provider*) durch ein Föderationskonzept zu einem globalen Modell vereinigt werden. Dazu wird, neben einer Infrastruktur für den Zugriff auf föderierte Kontextdaten, vor allem auch ein allgemeines Informationsmodell benötigt, das die Typen und Strukturen der enthaltenen Datenobjekte sowie den Zugriff auf diese Objekte beschreibt. Dieses als *Augmented World Model* bezeichnete allgemeine Kontextmodell liefert damit eine einheitliche Sicht auf die verteilten Kontextdaten. [LBBN04, DHN+04, NGS+01]

Augmented World Model Das Augmented World Model der Nexus-Plattform stellt ein objektorientiertes Informationsmodell für mobile ortsbasierte Anwendungen dar. Dabei beruht es auf der Idee, Objekte der realen Umgebung sowie zugehörige digitale Anreicherungen (*Augmentations*) abbilden zu können (vgl. Abbildung 3.14(a)). Um dies zu erreichen, bietet das Modell die Möglichkeit, reale Dinge, wie Gebäude, Räume, Straßen oder Personen und virtuelle Objekte, wie zum Beispiel *Virtual Information Towers* (VIT), abzubilden. Dabei dienen vor allem räumliche Informa-



(a) Reale und angereicherte Welt (b) Standard-Klassenschema

Abbildung 3.14.: Augmented World Model (nach (LBBN04, RFM+03, NGS+01))

tionen als Primärkontext und somit als Index auf die enthaltenen Daten. [LBBN04, NGS+01]

Das globale Augmented World Model ist dabei nicht explizit realisiert, sondern setzt sich aus lokalen Teilmodellen, den so genannten *Augmented Areas*, zusammen und ergibt sich erst durch deren Föderation als eine virtuelle Sicht auf die vorhandenen Kontextinformationen. Um eine Navigation im räumlichen Modell möglichst gut zu unterstützen, enthält es zusätzlich spezielle Graphobjekte, wie Knoten und Kanten, die mit realweltlichen Objekten verknüpft werden können. [LBBN04, NGS+01]

Realisiert ist dieses Basiskontextmodell in einem erweiterbaren Standard-Klassenschema (*Standard Class Schema*). Es umfasst 250 predefinierte Klassen beziehungsweise Typen von Kontextobjekten, die aus einer Use-Case-Analyse als relevant hervorgegangen sind (vgl. Abbildung 3.14(b)). Alle Dienste, die als Kontext-Provider an der Nexus-Systemumgebung teilnehmen möchten, müssen dieses grundlegende Modell unterstützen, sodass entsprechende Anwendungen jeden dieser Kontextkonten ansprechen können, um ihre Aufgabe zu erfüllen. Sollten Erweiterungen oder Spezialisierungen der Basisklassen zur Abbildung individueller Kontextinformationen notwendig sein, kann das Standard-Klassenschema von den einzelnen Kontext-Providern entsprechend ihrer Bedürfnisse erweitert und ergänzt werden. [LBBN04, NGS+01]

Um Kontextdaten des Augmented World Models zu erhalten, werden Anfragen an das System in einer einfachen ortsbezogenen Anfragesprache – der *Augmented World Query Language* (AWQL) – gestellt. Die Ergebnisse solcher Anfragen werden dann in einer einheitlichen, XML-basierte Spra-

che zwischen den Systemkomponenten der Nexus-Plattform und ihren Anwendungen übertragen. Diese als *Augmented World Modeling Language* (AWML) bezeichnete Auszeichnungssprache definiert allgemeine Sprachelemente, um Kontextobjekte mit ihren Attributen und Datentypen zu beschreiben und so zu serialisieren. [NGS+01]

Nexus-Plattform Die Architektur der Nexus-Plattform zum Verwalten des Augmented World Models gliedert sich in drei Ebenen (vgl. Abbildung 3.15): Auf der untersten Ebe befinden sich dabei die Dienstanbieter der einzelnen Augmented Areas zusammen mit den von ihnen verwalteten Kontextdaten. Auf der mittleren Schicht werden die einzelnen Kontexte der darunter liegenden Dienstschicht integriert und so zu einer virtuellen Sicht auf das Gesamtmodell vereinigt. Darüber hinaus enthält diese Föderationsschicht auch erweiterte Dienste, die auf dem Augmented World Model aufbauen. Auf oberster Ebene der Architektur liegen die Anwendungen, welche die Kontextdaten und Dienste der Nexus-Middleware nutzen und so kontextbasierte Dienste anbieten. [LBBN04]

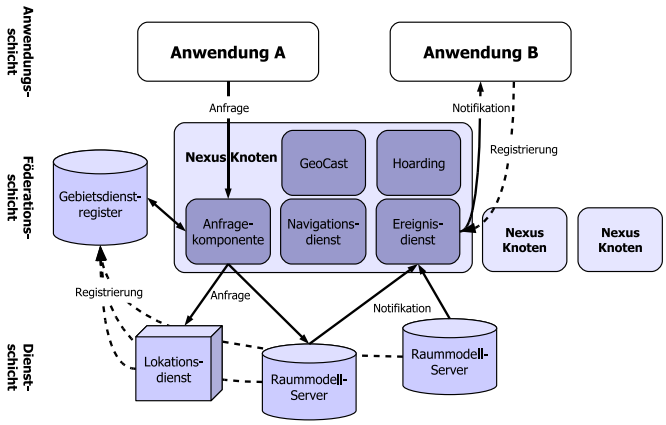


Abbildung 3.15.: Architektur der Nexus-Plattform (nach (DHN+04))

Als Basis der Nexus-Middleware fungieren Raummodell-Server (*Spatial Model Server*), welche die Kontextdaten einer oder mehrerer Augmented Areas verwalten. Sie enthalten dabei die statischen Objekte und Metadaten der einzelnen lokalen Kontextmodelle und stellen diese über eine einheitliche Schnittstelle (*Nexus Service Interface*) ihren Klienten zur Verfügung. Da Raummodell-Server lediglich statische Objekte speichern und verwalten sowie notwendige Funktionen für mobile Objekte fehlen (wie zum Beispiel die Unterstützung von hohen Aktualisierungsraten oder eine Übergabe (*Handover*) der Objekte von einem Management-

dienst zum nächsten), werden mobile Entitäten durch spezielle Lokationsdienste transient verwaltet. Sowohl die Raummodell-Server als auch die Lokationsdienste müssen sich im Gebietsdienstregister (*Area Service Register*) mit ihren Metadaten zu den von ihnen abgedeckten Gebieten und den vorgehaltenen Objekttypen registrieren, damit die Dienste der Föderationsschicht sie finden und abfragen können. [LBBN04, DHN+04, NGS+01]

Die Föderationsschicht der Nexus-Plattform bietet eine einheitliche und globale Sicht auf die Daten der Dienstschicht, indem sie zwischen den Anwendungen und den einzelnen Kontextdiensten vermittelt. Diese schicht besitzt die Aufgabe, Anfragen aus der Anwendungsschicht zu analysieren, sie gegebenenfalls in Teilanfragen aufzuteilen und die entsprechenden Raummodell-Server beziehungsweise Lokationsdienste weiterzureichen. Dazu greifen die Knoten dieser Schicht (*Nexus-Konten*) auf Informationen der Gebietsregister zu, um die für die Anfrage relevanten Teilkontexte und somit die verantwortlichen Server zu ermitteln. Die Ergebnisse der Teilanfragen werden zu einem Gesamtergebnis vereinigt und an die anfragende Anwendung als Resultat zurückgeliefert. Da die Nexus-Knoten selbst keine persistenten Daten verwalten, können sie beliebig oft repliziert werden und so eine Lastbalancierung im Gesamtsystem erreicht werden. [LBBN04, DHN+04, NGS+01]

Neben der einheitlichen Fähigkeit zur Abfrage des Augmented World Models unterstützen Nexus-Knoten auch höherwertige Dienste, die auf den föderierten Kontextdaten aufbauen (vgl. Abbildung 3.15). Dies ist zum Beispiel ein *Ereignisdienst*, der räumliche Ereignisse überwacht, diese gegebenenfalls zu komplexeren Ereignissen aggregiert und seine Abonnenten über das Eintreten der Ereignisse informiert. Daneben gibt es aber auch einen *Navigationsdienst*, der Routen zwischen einem Start- und Endpunkt berechnet oder einen Kommunikationsdienst (*GeoCast*), der einen Nachrichtenversand mit räumlicher Adressierung ermöglicht. [LBBN04, DHN+04]

3.3.3.3. Gaia-Middleware

Die an der *Universität von Illinois* entwickelte *Gaia-Middleware-Plattform* ist als Software-Infrastruktur zur Unterstützung der Entwicklung und des Einsatzes von Anwendungen in häuslichen Ubiquitous-Computing-Umgebungen entwickelt worden. In diesen als *Active Spaces* bezeichneten erweiterten physikalischen Räumen sollen Nutzer simultan mit unterschiedli-

chen Geräten interagieren, neue Geräte einfach und nahtlos in das vorhandene Umfeld integrieren sowie auf lokale und entfernte persönliche Daten komfortabel zugreifen können. Dabei sind Active Spaces folgendermaßen definiert: [RHC+02b, RHC+02a]

> *[...] An **Active Space** [is defined] as a physical space coordinated by a responsive context-based software infrastructure that enhances the ability of mobile users to interact and configure their physical and digital environment seamlessly.*
>
> [RHC+02b, Seite 65]

Die Gaia-Plattform ist als verteiltes Meta-Betriebssystem konzipiert, das Softwareeinheiten sowie heterogene und vernetzte Geräte einer physikalischen Umgebung koordiniert. Dabei erlaubt die verwendete Erweiterung des traditionellen Betriebssystemkonzepts eine unkomplizierte Verwaltung des Active Space und eine einfache Entwicklung von Anwendungen, da sich etablierte Methoden und Modelle übertragen lassen. Insgesamt bietet Gaia damit eine Abstraktion, in der die gesamte Umgebung als eine homogene programmierbare Einheit erscheint und nicht als eine Ansammlung individueller, heterogener Geräte. [RHC+02b]

Meta-Betriebssystem-Ansatz Das Gaia-Meta-Betriebssystem basiert auf einem verteilten Objektsystem, dass die Ressourcen und Dienste eines benutzerzentrierten Active Space verwaltet. Dabei ist es in drei Hauptkomponenten unterteilt: Dies sind der Gaia-Kernel, das Gaia-Anwendungsrahmenwerk und eine Verwaltungskomponente für Anwendungen im Active Space (vgl. Abbildung 3.16). [RHC+02b, RHC+02a]

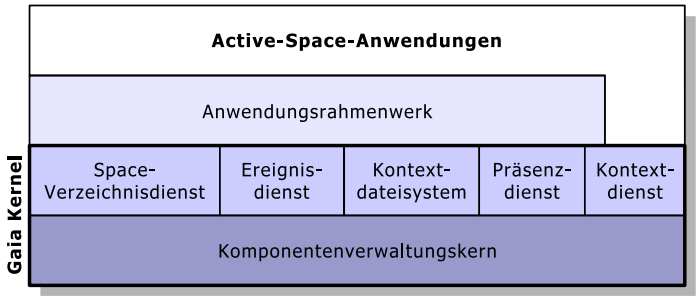


Abbildung 3.16.: Architektur der Gaia-Middleware (nach (RHC+02b))

Der *Gaia-Kernel* ist selbst wieder in ein Deployment- und Verwaltungssystem für verteilte Objekte sowie in grundlegende Dienste unterteilt, die von allen Anwendungen genutzt werden können. Dabei hat der *Komponen-*

tenverwaltungskern die Aufgabe, alle Komponenten der Gaia-Plattform dynamisch zu laden, zu erzeugen, zu transferieren beziehungsweise sie gegebenenfalls wieder zu löschen. Als asynchronen Kommunikationsdienst bietet der Kernel des Systems einen *Ereignisdienst* zur Kommunikation nach dem Publish/Subscribe-Modell an, der Nachrichten über Ereignisse im Active Space verbreiten kann (vgl. 3.2.2). Neben einem *Verzeichnisdienst*, der Informationen über alle Soft- und Hardwarekomponenten der Umgebung bereitstellt, steht den Anwendungen auch ein *Präsenzdienst* zur Verfügung, über den Informationen zu digitalen und physikalischen Einheiten bezogen werden können, die selbst keine native Gaia-Komponente darstellen – wie beispielsweise der Nutzer selbst. Um es Anwendungen zu ermöglichen, ihr Verhalten an den Kontext des Active Space anzupassen, enthält die Gaia-Plattform einen *Kontextdienst*, der Basis- und abgeleitete Kontextinformationen entweder proaktiv über Ereignisnachrichten oder reaktiv über gezielte Abfragen bereitstellt. Das *Kontextdateisystem* verbindet Kontextinformationen mit dem traditionellen Dateisystemmodell, um die Mobilität von Nutzern zu unterstützen, die Heterogenität der am Active Space beteiligten Geräte zu überwinden und eine komfortable Verwaltung von lokalen und entfernten Daten zu erreichen. Das Dateisystem ist dabei hierarchisch organisiert und ermöglicht das automatische Einbinden von persönlichen Daten und Dateien sowie deren logische Adressierung, um von physikalischen Adressen unabhängig zu sein. Zudem bietet es die automatische Konvertierung der Daten zwischen unterschiedlichen Formaten an, die jeweils durch den Kontext des Nutzers und des Gerätes bestimmt werden können. [RHC+02b, RHC+02a]

Das *Anwendungsrahmenwerk* baut auf einem Anwendungsarchitekturmodell auf, das eine Erweiterung des klassischen MVC-Modells[3] darstellt. Es führt dabei das neue funktionale Konzept eines Koordinators ein, um die Bindung der Anwendungskomponenten im Active Space exportieren und verwalten zu können. Damit ist es möglich, die einzelnen Komponenten der Anwendung auf unterschiedlichen Geräten auszuführen und so eine verteilte und dynamisch rekonfigurierbare Anwendung zu schaffen. Um die Anwendungskomponenten auf die jeweiligen Geräte abzustimmen, führt das Anwendungsrahmenwerk der Gaia-Plattform Dokumente mit zusätzlichen Metadaten ein, mit deren Hilfe die Anwendung auf die

[3] Das Model-View-Controller-Architekturmuster (MVC) strukturiert den Aufbau von Anwendungen in drei lose gekoppelte Komponenten, die für die Verwaltung des Datenmodells, die (grafische) Repräsentation der Anwendung und die Verarbeitung von Benutzerinteraktionen verantwortlich sind.

Konfigurationen verschiedener Active Spaces abgebildet werden können. [RHC+02b, RHC+02a]

Auf der *Active-Space-Anwendungsschicht* der Gaia-Architektur sind die ausführbaren Anwendungen des Gaia-Systems angesiedelt. Sie besitzt dabei Funktionalitäten zum Registrieren, Verwalten sowie Betreiben der Anwendungen, welche selbst wiederum die Dienste des Gaia-Kernels nutzen. [RHC+02b, RHC+02a]

Gaia-Kontextstruktur Das Kontextmodell der Gaia-Plattform ist mit Hilfe der *Prädikatenlogik erster Ordnung* definiert. Basiselemente sind dabei Low-Level-Kontextdaten, die als Prädikat mit vier Werten modelliert werden und über folgende Struktur definiert sind:

Context(<ContextType>, <Subject>, <Relater>, <Object>)

Hierbei gibt der *ContextType* die Art der Kontextinformation an, die durch das Prädikat beschrieben wird. Die Entität, auf die sich die Information bezieht, wird durch den *Subject*-Eintrag des Tupels angegeben. Das *Object* gibt im Prädikat den eigentlichen Wert an, der mit der Entität assoziiert ist. Da der Wert und die Entität in unterschiedlichen Beziehungen zueinander stehen können, kann durch den *Relater* die aktuelle Relation zwischen beiden angegeben werden. Dabei kann die Beziehung vergleichende Operationen, Verben oder Präpositionen umfassen. [RCRM02]

Operation	Beschreibung
$\wedge$	Konjunktion (UND-VERKNÜPFUNG)
$\vee$	Disjunktion (ODER-VERKNÜPFUNG)
$\longrightarrow$	Implikation (SCHLUSSFOLGERUNG)
NOT	Negation
$\forall$	Allquantor (FÜR ALLE GILT)
$\exists$	Existenzquantor (FÜR MIND. EINEN GILT)

Tabelle 3.2.: Operationen auf Gaia-Kontextstrukturen

Neben den als atomares Prädikat beschriebenen Kontextinformationen lassen sich im Gaia-Kontextmodell auch abgeleitete High-Level-Kontextdaten durch das Anwenden von Booleschen Operationen und das Verwenden von Quantoren konstruieren (vgl. Tabelle 3.2). So kann beispielsweise aus den Kontextdaten über die Anzahl der Personen in einem Vortrags-

raum und einer laufenden Präsentationssoftware auf den Kontext eines Vortrages geschlossen werden. Durch die Verwendung von Variablen und Quantoren bei der Definition von Kontextprädikaten ist es zudem möglich, Aussagen auf Mengen von Entitäten auszudehnen und so erweiterte Kontextaussagen zu formulieren. [RCRM02, RHC+02a]

Gaia für mobile Systeme Die Gaia-Middleware für Active Spaces ist für den Einsatz in Umgebungen mit ressourcenreichen Geräten konzipiert und hat dementsprechend Schwächen in der Integration von eher ressourcenarmen mobilen Geräten. Um dieses Manko zu kompensieren sind zwei Gaia-Erweiterungen für mobile Systeme entwickelt worden. Eine dieser Erweiterungen verwendet dabei ein *Proxykonzept*, um von mobilen Geräten auf die Gaia-Middleware zuzugreifen, die andere hingegen adaptiert das Konzept der Active Spaces und die Middleware-Plattform selbst, um so *Personal Active Spaces* zu unterstützen. [CBAC05]

Mobile Gaia ist wie die Standardumgebung eine dienstbasierte Middleware, die jedoch gezielt für ubiquitäre Ad-hoc-Systeme – den *Personal Active Spaces* – konzipiert ist. Diese Umgebungen haben eine kleinere Ausdehnung und bestehen aus einem Cluster von mobilen Geräten, die untereinander vernetzt sind und sich gegenseitig Ressourcen zur Verfügung stellen. Jeder Personal Active Space besitzt einen *zentralen Koordinatorknoten*, der den Cluster verwaltet und dabei neue Geräte (*Klienten*) in seiner Umgebung erkennt und einbindet sowie Basisdienste für das verteilte Ressourcenmanagement bereitstellt. Dabei ist die Mobile-Gaia-Plattform so konzipiert, dass Dienste – je nach Rolle und Bedarf – dynamisch geladen und eingebunden werden können und so eine Abstimmung auf die Möglichkeiten des mobilen Gerätes erfolgen kann. [CACM05]

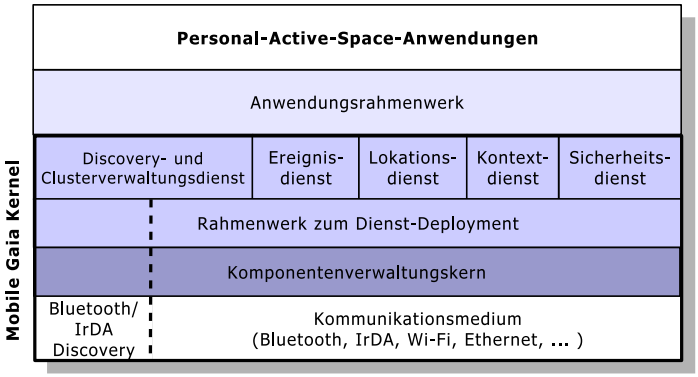


Abbildung 3.17.: Mobile-Gaia-Architektur (nach (CACM05))

Der Aufbau der Mobile Gaia Middleware besteht ähnlich wie die Stan-

dardplattform aus einem Kernel, einem Anwendungsrahmenwerk und einer Anwendungsschicht (vgl. Abbildung 3.17). Jedoch wird die Architektur um einen Dienst zum Laden und Verwalten der Kernel-Basisdienste ergänzt, um so dynamische Rekonfigurationen vornehmen zu können. Der *Discovery- und Clusterverwaltungsmanager* der Architektur hat die Aufgabe, andere Geräte in der Nachbarschaft zu finden und die Teilnahme am Cluster zu organisieren. Die Kommunikation zwischen den Geräten wird wie in der Standardvariante der Middleware mittels eines *Ereignisdienstes* realisiert. Um Orts- und Kontextinformationen zu nutzen, bietet die Mobile-Gaia-Plattform jeweils einen *Lokations- und Kontextdienst*. Die erhöhten Sicherheitsanforderungen eines Personal Active Spaces werden durch einen *Sicherheitsdienst* umgesetzt, der für die Authentifizierung der Teilnehmer eines Clusters und die Zugriffskontrolle auf die bereitgestellten Ressourcen verantwortlich ist. [CACM05]

Einen anderen Weg geht der *Gaia Microserver*, der ressourcenarme Geräte mit Hilfe eines Proxy-Konzepts in Active Spaces einbindet. Durch diese Indirektion ist es mobilen Geräten möglich, mittels einer standardisierten Schnittstelle als Aus- und Eingabegeräte am Gaia-System teilzunehmen. Der Microserver dient so als Interoperabilitätsdienst, der die Unzulänglichkeiten mobiler Geräte mit geringen technologischen Fähigkeiten überwinden kann. [CBAC05]

3.3.3.4. Blackboard-based Context Framework

Ziel des *Blackboard-based Context Frameworks* ist es, Anwendungen für mobile Geräte um kontextbezogene Fähigkeiten zu erweitern, ohne die Anwendungen selbst ändern zu müssen. Dazu wird die Kontextgewinnung und -verwaltung konsequent von den Anwendungen getrennt sowie die kontextbezogene Anpassung durch die regelbasierte Aktivierung von Anwendungsaktionen und das Auslösen von Systemereignissen durchgeführt. Dabei werden die Regeln, welche die Beziehung zwischen Kontextdaten und den daraus resultierenden Aktionen angeben (*Kontext-Aktions-Regeln*), durch den Nutzer selbst erzeugt und durch das Kontextrahmenwerk lediglich verwaltet und ausgewertet. [KMS+05]

Architektur des Rahmenwerkes Die Architektur des Blackboard-based Context Frameworks (vgl. Abbildung 3.18) besteht aus einem Kernsystem zum Akquirieren und Verwalten von Kontextdaten, das mit der Fähigkeit erweitert wird, beliebige Anwendungen eines mobilen Gerätes kontextba-

siert zu steuern. Dabei bildet der auf dem Blackboard-Modell (vgl. Ab-

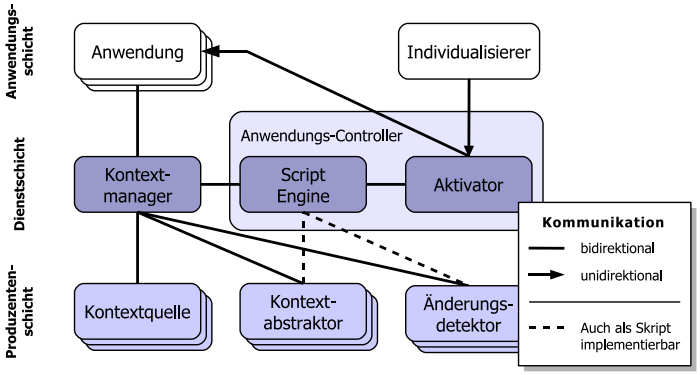


Abbildung 3.18.: Architektur des blackboard-basierten Rahmenwerkes (nach (Kor05))

schnitt 3.2.3) basierende *Kontextmanager* die zentrale Komponente des Systems. Dieser bezieht Kontextdaten von beliebigen internen oder externen *Kontextquellen* und stellt diese seinen Klienten einheitlich zur Verfügung. Der Kontextmanager stellt somit die gebündelte Kontextdatenbasis dar, kann aber auch zur Kommunikation der Klienten untereinander genutzt werden. Der Zugriff auf die Kontextinformationen kann dabei entweder durch einzelne Suchanfragen (*Pull-Semantik*) oder durch das Registrieren von Benachrichtigungswünschen (*Publish/Subscribe-Semantik* – vgl. Abschnitt 3.2.2) vorgenommen werden. Ergänzt wird der Kontextmanager auf der Dienstschicht durch den Anwendungs-Controller, der für das kontextbasierte Steuern von ursprünglich nicht-kontextsensitiven Anwendungen verantwortlich ist. Hieraus ergibt sich also eine dritte Art der Kontextnutzung auf der Anwendungsschicht. [Kor05, KMS+05, KMK+03]

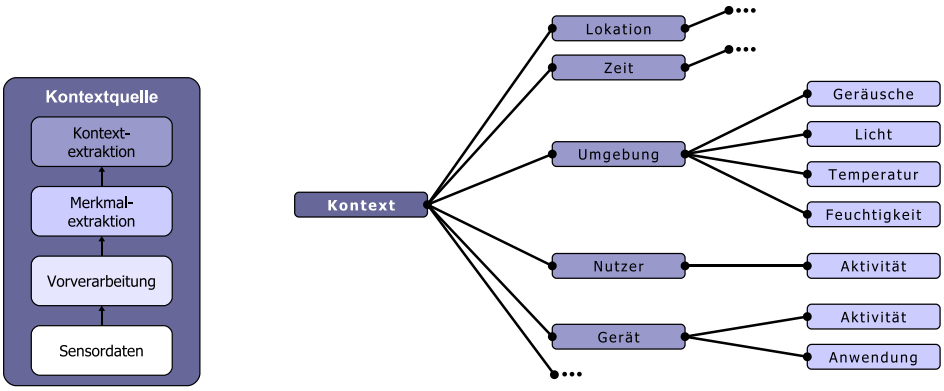
Der *Anwendungs-Controller* ist als Empfänger von Kontextdaten beim Kontextmanager registriert und führt auf dieser Basis in Verbindung mit vom Anwender definierten Regeln entweder direkte Aufrufe an den Schnittstellen von Anwendungen aus oder generiert Systemereignisse, die das Betriebssystem der Host-Plattform weiterverarbeitet. Der Aufbau dieser Controller-Komponente gliedert sich dabei in eine Script Engine und einen Aktivator. [Kor05]

Die regelbasierte Inferenzmaschine (*Script Engine*) der Controller-Komponente führt arithmetische und logische Operationen auf Kontextdaten aus, die in der XML-basierten Sprache des *Context Exchange Protocols*[4] (CEP) verfasst sind. Jede Regel definiert dabei, welche Kontextdaten be-

[4] Das Context Exchange Protocol definiert, wie Kontextdaten und zugehörige Operationen dargestellt und ausgetauscht werden können. [Lak03]

nötigt werden, wie sie verarbeitet werden und welche Aktionen ausgeführt werden, wenn die Bedingungen für die Regel eintreten. Dabei können neben der direkten oder indirekten Manipulation von Anwendungen auch abgeleitete Kontextdaten generiert und an den Kontextmanager übergeben werden. Treten die Bedingungen zur Ausführung einer Regel ein, können auch diejenigen mittels einer Nachricht informiert, welche die entsprechende Regel an die Script Engine übergeben haben. Dies ist im Allgemeinen der *Aktivator*, der die vom Benutzer generierten Regeln verwaltet und zur Ausführung bringt. Auf solch eine Benachrichtigung hin führt dann der Aktivator die definierten Funktionsaufrufe an der Anwendungsschnittstelle aus oder generiert entsprechende Systemereignisse. Diese Aufteilung der Funktionalität ist deshalb notwendig, da die Aktivator-Komponente – im Gegensatz zur Inferenzmaschine – nicht plattformunabhängig sein kann und deshalb an die jeweilige Host-Plattform angepasst werden muss. [Kor05, KMS+05]

Gewinnung einheitlicher Kontextdaten Wie in jeder kontextbezogenen Middleware kommt auch im Blackboard-based Context Framework der Gewinnung von qualitativ hochwertigen und semantisch einheitlichen Kontextdaten ein hoher Stellenwert zu. Dazu nutzt das Rahmenwerk ein einheitliches aber erweiterbares Kontextvokabular sowie eine uniforme Architektur zur Integration von lokalen oder externen Rohdatenquellen.



(a) Quellknotenstruktur (b) Kategorien des erweiterbaren Kontextvokabulars

Abbildung 3.19.: Kontextdatengewinnung (nach (Kor05, KMK+03))

Kontextquellen sind Wrapper-Komponenten, die Datenquellen kapseln, aufbereiten und einheitlich über den Kontextmanager an Anwendungen und andere Komponenten des Systems weitergeben, zum Beispiel zum Gewinnen von abstrakten und höherwertigen Kontextdaten (*Kontextab-*

straktoren). Dabei durchlaufen Sensordaten innerhalb des Quellknotens einen eigenen Prozess, damit die Kontextdaten auf dem richtigen Abstraktionsniveau und mit einer für das System beherrschbaren Frequenz bereitgestellt werden (vgl. Abbildung 3.19(a)). Dazu werden die Sensorrohdaten zunächst durch eine *Vorverarbeitung* diskretisiert – es werden also mehrere Messwerte zu einem Ergebnis für ein Messintervall aggregiert. Durch das Festlegen der Intervallgröße lässt sich so die Frequenz beeinflussen, mit der sich Änderungen auf das Kontextsystem auswirken. In der anschließenden Phase der *Merkmalsextraktion* wird der gewonnene Messwert an einen Eintrag in einer Ontologie gebunden und erhält so eine Bindung zu einem Begriff einer Domäne. Dabei kann zum Beispiel ein nummerischer Messwert in einen abstrakten Begriff überführt werden. Die abschließende *Kontextextraktion* generiert ein atomares Kontextobjekt, das die abstrakte Kontextinformation entsprechend des *erweiterbaren Kontextvokabulars* (vgl. Abbildung 3.19(b)) zugreifbar macht, sodass dieses an das Kontextmanagement übergeben werden kann. [Kor05, KMS+05, KMK+03]

3.4. Implikationen für diese Arbeit

Im vorstehenden Kapitel ist dargelegt, dass gerade auch in mobilen Umgebungen der Einsatz von Middleware-Plattformen mit ihren Basisdiensten sowie Protokollen und Sprachen eine sinvolle Abstraktion zur Anwendungsimplementierung und -ausführung darstellen. Hierbei gehen jedoch die Anforderungen, die an die Middleware-Plattformen gestellt werden, über die in klassischen Verteilten Systemen hinaus, da die intrinsischen Eigenschaften mobiler Systeme Berücksichtigung finden müssen. Dabei ist von besonderer Bedeutung, inwieweit die Verteilungstransparenz zugunsten eines Bewusstseins über die mobile Umgebung aufgegeben wird.

Um bei der Realisierung von auf mobile Systeme abgestimmten Middleware-Plattformen eine zeitliche Entkopplung der Kommunikation zu erreichen, werden häufig asynchrone Kommunikationsparadigmen verwendet, die unter den unzuverlässigen Bedingungen einer mobilen Umgebung besonders vorteilhaft sind. Dabei werden neben der nachrichtenorientierten Kommunikation über Warteschlangensysteme auch ereignisbasierte und Tuple-Space-basierte Systeme verwendet. Die auf diesen Paradigmen aufbauenden kommunikationsorientierten Middleware-Systeme bilden meist die Grundlage für durch zusätzliche Dienste erweiterte anwendungsorientierte Plattformen. Dabei integrieren diese vor allem Informationen über

den Kontext des mobilen Gerätes und unterstützen so Anwendungen, die auf den Middleware-Systemen aufbauen, bei ihrer Anpassung an die sich stetig verändernden Bedingungen der mobilen Umgebung.

Dieser Kontextbezug von Middleware-Plattformen – also dem Zugänglichmachen von Daten über die Veränderung der Situation, in der sich eine Entität befindet – erlaubt es, neuartige Applikationen zu entwickeln, die ihre Datenselektion, ihre Präsentation von Informationen oder ihre Handlungen entsprechend den erkannten Situationen anpassen. Die zur Abbildung der Eigenschaften der mobilen Umgebung verwendeten Kontextmodelle reichen dabei von anwendungsabhängigen und somit begrenzten Modellen über universell einsetzbare abstrakte Modelle hin zu föderierten Konzepten, die ein globales Weltmodell anstreben. Jedoch bleiben die Anwendungen, die auf der Basis dieser bestehenden Systeme entwickelt werden, meist eher monolithisch und ad-hoc-statisch strukturiert und unterstützen so den Nutzer nur bei eher kurzzeitigen und einfachen Aufgaben.

Um eine Integration auch langandauernder und strukturierter Aufgaben in ein solches Umfeld zu ermöglichen, wird im nächsten Kapitel untersucht, wie solche komplexen Aufgaben modelliert und ausgeführt werden können. Dazu werden die aus dem *Service Oriented Computing* stammenden Konzepte der Dienst- und Prozessorientierung eingeführt und auf exemplarische Umsetzungen dieser Paradigmen eingegangen.

4. Dienste und Prozesse zur metasprachlichen Beschreibung komplexer Aufgaben

Mobile Umgebungen sind stark durch die Heterogenität der beteiligten Geräte und Softwaresysteme geprägt (vgl. Abschnitt 2.3.2). Um in solch einem Umfeld komplexe beziehungsweise strukturierte Aufgaben für das Gesamtsystem beschreiben und ausführen zu können, müssen diese zunächst auf eine möglichst abstrakte Art definiert werden, welche die Kerneigenschaften der Aufgaben von den zur Ausführung benötigten technologischen Merkmalen trennt. Damit ist es dann möglich, zur Laufzeit zu entscheiden, welche der verfügbaren Ausführungseinheiten im konkreten System einen Teil der komplexen Aufgabe beziehungsweise die gesamte Funktionalität ausführen können. [Hol04]

Eine solche Abstraktion bietet unter anderem das Paradigma des *Service-Oriented Computing*, das Dienste als Basiselemente für die Konstruktion von Anwendungen verwendet. Da Dienste selbstbeschreibend und plattformungebunden sind, ermöglichen sie die schnelle und einfache Komposition von verteilten Anwendungen gerade auch in mobilen Umgebungen. [Per06, Pap03, PG03]

4.1. Interoperabilität durch Dienstorientierung

Die dienstorientierte Sichtweise auf Softwaresysteme greift das im *Software Engineering* bekannte Modularisierungs- und Black-Box-Prinzip auf. So wie Module, Objekte oder Komponenten einzelne abgrenzbare Softwareeinheiten innerhalb eines Programms beschreiben, stellen Dienste eine dezidierte *Business-Funktion* dar, die einzeln oder in Kombination mit anderen Diensten Verwendung finden kann. Dabei soll durch den Einsatz von standardisierten Schnittstellen und Zugriffsmethoden eine Integration der Systemkomponenten erreicht werden, um eine Virtualisierung der verfügbaren Ressourcen und damit eine Automatisierung sowie Optimierung von Arbeitsabläufen zu ermöglichen. Um dies zu erzielen, sollen Dienste möglichst *technologieneutral* sein, was bedeutet, dass sie mit Hilfe von offenen Standards aufgerufen werden, die auf möglichst allen Systemen verfügbar sind. Zudem sollen Dienste eine *lose Kopplung* bieten, bei der

Aufrufer keine Kenntnis über die interne Struktur des Dienstes besitzen müssen, um diesen zu verwenden. Diese lose Kopplung ist zudem eine der Voraussetzungen des *dynamischen Bindens*, bei dem ein Dienst erst zur Laufzeit automatisch gesucht, gefunden und angesprochen wird. Darüber hinaus sollen Dienste *ortstransparent* sein (vgl. Abschnitt 2.3.1), damit Klienten – unabhängig von ihrem eigenen Standort – einen benötigten Dienst lokalisieren und ansprechen können. [DJMZ05, KL04, Pap03]

Dienstorientierte Systeme bauen im Allgemeinen auf der *Service-Oriented Architecture* (SOA) auf, die angibt, wie Anwendungen und Infrastrukturkomponenten organisiert sein sollten, damit sie als interagierende Dienste Verwendung finden können. Die SOA stellt dabei jedoch keine konkrete Technologie dar – sie ist vielmehr eine Abstraktion, die wesentliche Aspekte des Designs einer verteilten Dienstearchitektur in einer logischen Sichtweise vereinigt. Dabei beschreibt das Grundprinzip der Service-Oriented Architecture die Interaktion zwischen den abstrakten Rollen des Dienstanbieters und des Dienstnutzers (*Klienten*) sowie zwischen der vermittelnden Rolle des Dienstverzeichnisses. Eines der primären Ziele einer solchen dienstorientierten Architektur ist es, dass Softwarekomponenten einen automatisierten Zugriff auf Dienste erhalten und so eine *Maschine-zu-Maschine-Kommunikation* ermöglicht wird. Insgesamt soll so ein System aus vernetzten, lose gekoppelten und kommunizierenden Diensten entstehen. [Fos05, DJMZ05, Pap03]

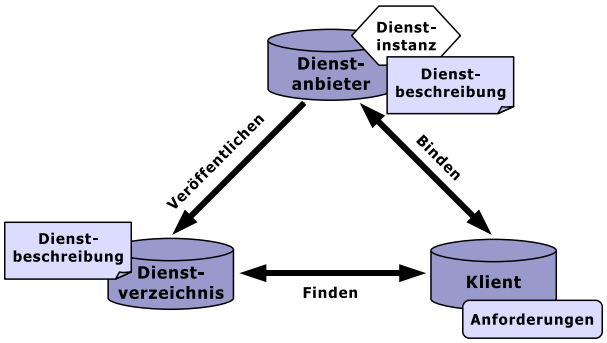


Abbildung 4.1.: Elementare SOA-Komponenten (nach (Per06, Pap03))

4.1.1. Das grundlegende SOA-Modell

Das grundlegende SOA-Modell beschreibt, welche Beziehungen und Interaktionen zwischen Dienstanbietern, -nutzern und -verzeichnissen in einem dienstorientierten System bestehen (vgl. Abbildung 4.1). Dabei führen die beteiligten Komponenten die grundlegenden Operationen des *Veröffentli-*

chens eines Dienstes, des *Findens* von geeigneten Dienstinstanzen und des *Bindens* respektive Ausführen des gewählten Dienstes aus. [Pap03, PG03]

Dienstanbieter Dienstanbieter stellen Business-Funktionen mittels einer Software-Komponente als Dienst bereit. Damit der Dienst von anderen gefunden und genutzt werden kann, erstellt der Dienstanbieter eine Dienstbeschreibung, die er beim Dienstverzeichnis registrieren, aktualisieren und gegebenenfalls wieder löschen kann. Durch die Dienstbeschreibung werden die Fähigkeiten des Dienstes, seine Schnittstelle und sein Verhalten sowie Qualitätsaspekte spezifiziert. Dienstanbieter müssen dabei nicht jeden von ihnen angebotenen Dienst selbst implementieren. Er kann vielmehr selbst wieder Dienste nutzen, diese kapseln oder zu höherwertigen Diensten aggregieren. [DJMZ05, Pap03, PG03, TP02]

Klient Nutzer beziehungsweise Klient eines Dienstes ist derjenige Partner eines Aufrufes, der eine bereitgestellte Business-Funktion benötigt und deshalb einen Dienst nutzen möchte. Um einen geeigneten Dienst zu finden, verwenden die Klienten im Allgemeinen die im Verzeichnis veröffentlichten Informationen und nutzen die enthaltene Schnittstellenbeschreibung, um den Dienst zu binden. Dazu werden offene Standards sowie Taxonomien und Ontologien benötigt, mit der die Informationen im Verzeichnis beschrieben und die Kommunikationsakte zwischen Dienst und Klient durchgeführt werden. [DJMZ05, Pap03, TP02]

Dienstverzeichnis Ein Dienstverzeichnis stellt ein Register dar, in dem Dienstanbieter ihre Dienstbeschreibungen veröffentlichen und in denen Klienten nach Diensten, die eine benötigte Business-Funktion bieten, suchen können. Die Klienten nutzen dabei die im Verzeichnis erhaltenen Daten, um den Dienst auszuwählen, zu binden und dann anzusprechen. Die Suche nach einem Dienst sollte dabei selbst wieder in Form eines Dienstaufrufes geschehen, damit eine einheitliche Aufrufstruktur und damit kein zusätzlicher Overhead entsteht. [DJMZ05, Pap03, TP02]

Dienstanbieter und Klient sind jedoch nur logische Konstrukte, die sich auf die Rolle des Kommunikationspartners in einem einzelnen Dienstaufruf beziehen. Anbieter können durchaus selbst wieder zu Klienten werden, wenn sie für die Erfüllung ihrer Funktionalität andere Dienste in Anspruch nehmen. [Pap03]

4.1.2. Web Services

Die Service-Oriented Architecture stellt zunächst ein implementierungsunabhängiges und somit abstraktes Modell einer Software-Architektur dar. Das Konzept der *Web Services* ist hingegen ein konkreter Ansatz, der die Anforderungen einer SOA erfüllt und eine weite Verbreitung gefunden hat. [DJMZ05] Hierbei kann ein Web Service abstrakt wie folgt definiert werden:

> *A **Web Service** is a specific kind of service that is identified by a URI[a], whose service description and transport utilize open Internet standards.*
>
> [PG03, Seite 27]

[a] Uniform Resource Identifier

Im Umfeld von Web Services haben sich für die Schlüsselaufgaben des SOA-Modells einzelne Standards durchgesetzt, die jeweils einen bestimmten Aufgabenbereich bei der Kommunikation abdecken (vgl. Abbildung 4.2). Interaktionen zwischen Web Services und ihren Klienten werden typischerweise über ein bestehendes Transportprotokoll mit Hilfe von *SOAP-Aufrufen* realisiert, die XML-formatierte Daten enthalten. Die Schnittstelle eines Dienstes wird in der *Web Service Description Language* (WSDL) ausgedrückt und mit Hilfe des UDDI-Standards (*Universal Description, Discovery, and Integration*) bei einem Verzeichnisdienst registriert, wo sie von potenziellen Klienten eines Dienstes wieder gefunden werden kann. Rund um diesen Basisprotokollstapel haben sich ergänzende Protokolle und Standards gebildet, welche zum Beispiel *Transaktionen*, die *Koordination* von Diensten oder *Sicherheitsaspekte* hinzufügen. Die Verwendung dieser einfachen und offenen Standards hat dabei zu der weiten Verbreitung und Dominanz von Web Services als Umsetzung des SOA-Modells geführt. [Ham05, DJMZ05, PG03, GGKS02]

SOAP *SOAP*[1] ist ein XML-basiertes Protokoll für den Nachrichtenaustausch zwischen Kommunikationspartnern und legt die Verarbeitungsvorschriften und die Grobstruktur der ausgetauschten Nachrichten fest. Dabei wird im Allgemeinen HTTP als Transportprotokoll verwendet, obwohl dies nicht vorgeschieben ist und auch andere Technologien, wie FTP oder nachrichtenorientierte Kommunikationsformen (vgl. Abschnitt 3.2.1), benutzt werden können. [KL04, TP02, Cer02]

[1] Heutzutage ist SOAP ein Eigenname – ursprünglich stand es als Akronym für *Simple Object Access Protocol*.

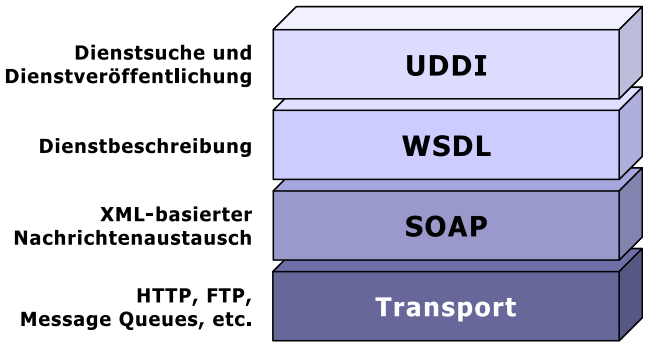
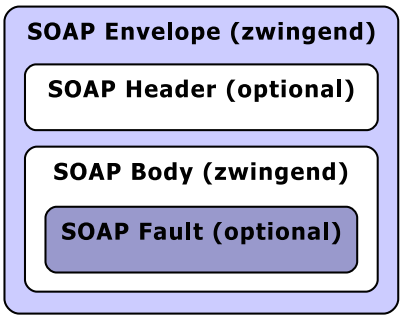


Abbildung 4.2.: Kernstandards des Web-Service-Protokollstapels

SOAP strukturiert eine Nachricht als einen in XML kodierten Umschlag (*Envelope*) mit einem optionalen Kopfteil (*Header*) und einem zwingend vorhandenen Nutzdatenteil (*Body*) (vgl. Abbildung 4.3). Im Kopfteil werden anwendungsabhängige Kontrollinformationen angegeben, die zum Beispiel für die Authentifizierung, das Transaktionsmanagement oder für Abrechnungszwecke genutzt werden können. Das Verarbeitungsmodell einer SOAP-Nachricht sieht dabei vor, dass Knoten, die zwischen Sender und Empfänger der Nachricht liegen – so genannte *Intermediaries* –, lediglich auf diesen Header-Daten arbeiten, ihnen Informationen hinzufügen oder löschen. Der Body der SOAP-Nachricht enthält die eigentlichen Nutzdaten, die mit der Nachricht übertragen werden sollen. Dies kann zum Beispiel ein *Remote Procedure Call* sein, in dem die Methoden und Argumente eines Dienstaufrufes enthalten sind. Zusätzlich kann der Body im Fehlerfall zusätzliche *Fault-Elemente* enthalten, die das Format von Fehlermeldungen zwischen Sender und Empfänger einheitlich regeln. [KL04, TP02, Cer02]



(a) Struktur

```
<?xml version="1.0">
<env:Envelope xmlns:env =
"http://www.w3c.org/soap-envelope">
  <env:Header> ... </env:Header>*
  <env:Body>
    ...
    <env:Fault> ... </env:Fault>*
  </env:Body>
</env:Envelope>
```

(b) XML-Kodierung

Abbildung 4.3.: SOAP-Nachrichten (nach (KL04, Cer02))

WSDL – Web Services Description Language Um die Schnittstelle eines Web Services zu definieren, wird im Allgemeinen die *Web Services Description Language* (WSDL) verwendet, die es erlaubt, ein- und ausgehende Nachrichten zu spezifizieren und diese anschließend zu Operationen zusammenzufassen. Mit der WSDL-Beschreibung werden die Fragen nach dem *was* der Dienst bietet, *wo* er sich befindet und *wie* er aufgerufen werden kann beantwortet. [KL04, TP02, Cer02]

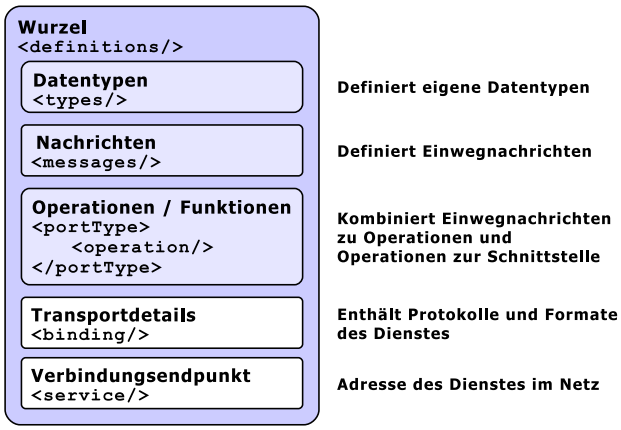


Abbildung 4.4.: WSDL-Struktur

So wie SOAP-Nachrichten werden auch WSDL-Schnittstellenbeschreibungen als XML-Dokumente kodiert. Dabei wird durch eine Separation der abstrakten Definition von Typen, Nachrichten und Operationen sowie der instanzspezifischen Informationen zum Binden und zum Netzwerkzugriff eine Wiederverwendung der Schnittstelleninformationen ermöglicht. Insgesamt wird der Dienst im WSDL-Dokument durch folgende Elemente beschrieben (vgl. Abbildung 4.4): [DJMZ05, TP02]

Das Wurzelelement und damit der Rahmen des WSDL-Dokumentes ist durch das `definitions`-Element bestimmt. Es definiert den Namen des Web Services und deklariert die Namensräume der im Dokument verwendeten Standards. Alle selbst definierten Typen, die in der Dienstbeschreibung benutzt werden, müssen zunächst im `types`-Element definiert werden. Mit Hilfe des Elements mit der Bezeichnung `message` werden alle zur Kommunikation mit dem Dienst notwendigen Nachrichten beschrieben. Jede dieser Nachrichten beschreibt dabei lediglich eine Einwegnachricht, die eine einzelne Anfrage oder Antwort repräsentiert. Diese einzelnen Nachrichten werden durch `operation`-Elemente zu den einzelnen Operationen des Dienstes zusammengefasst und diese im `portType`-Element zur abstrakten Schnittstelle aggregiert. [Ham05, Cer02, CCMW01]

Die Informationen zur abstrakten Schnittstelle werden durch Angaben im `binding`-Element, das die vom Dienst unterstützten Protokolle angibt, sowie um die Netzwerkadresse, unter welcher der Dienst angesprochen werden kann (`service`-Element), ergänzt. Dabei bietet WSDL zwar eine integrierte Unterstützung zur Definition von SOAP-Diensten an, ist jedoch nicht auf dieses Protokoll zum Nachrichtenaustausch beschränkt. [DJMZ05, Ham05, Cer02]

UDDI – Universal Description, Discovery, and Integration Die Aufgabe des vom SOA-Modell geforderten Verzeichnisdienstes wird im grundlegenden Web-Service-Protokollstapel durch den UDDI-Standard (*Universal Description, Discovery, and Integration*) realisiert. Der Standard definiert neben einer technischen Spezifikation zum Aufbau eines verteilten Verzeichnisses auch Standardmethoden, um Dienste im Verzeichnis zu veröffentlichen und zu suchen. Dabei hat das UDDI-Verzeichnis insgesamt die funktionale Aufgabe, Daten und Metadaten über Web Services abzubilden und zu verwalten. [OAS04, Cer02]

Über die interne Implementierung eines Verzeichnisses für Web Services macht die UDDI-Spezifikation keine konkreten Aussagen – sie definiert lediglich einzelne Komponenten, deren Beziehungen und Kommunikationsprotokolle sowie die Schnittstelle, über die auf das System zugegriffen wird. Konsequenterweise ist diese Schnittstelle selbst wieder als Web Service definiert, sodass sich das Verzeichnis nahtlos in die Web-Service-Landschaft einfügt. [OAS04]

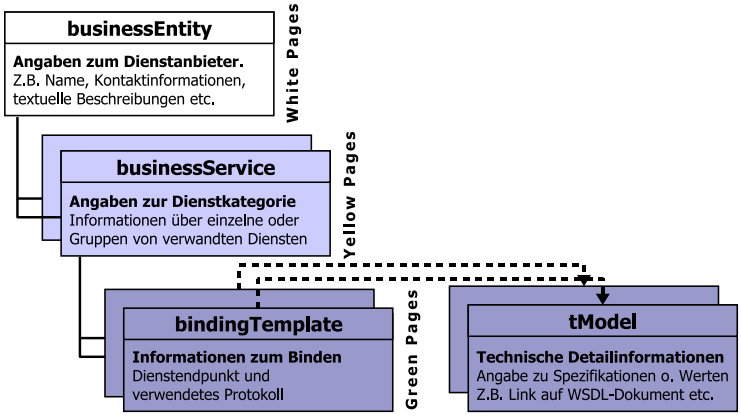


Abbildung 4.5.: UDDI-Datenmodell (nach (OAS04))

Die Daten und Metadaten, die in einem UDDI-Verzeichnis enthalten sind, werden in drei Kategorien unterteilt, die jeweils unterschiedliche

Aspekte eines Dienstes beschreiben und im UDDI-Datenmodell als *White*, *Yellow* und *Green Pages* bezeichnet werden (vgl. Abbildung 4.5). Generelle Angaben zu einem Anbieter, der einen Web Service veröffentlicht, werden in den White Pages in Form von `businessEntity`-Einträgen hinterlegt und beschreiben unter anderem den Namen des Anbieters und seine Kontaktinformationen. In den Yellow Pages hingegen werden allgemeine Angaben zur Klassifizierung einzelner oder mehrerer verwandter Dienste gespeichert. Die so genannten `businessService`-Einträge beschreiben also die Geschäftsfunktion, die ein Dienst anbietet. Die Informationen, die zum Binden eines Dienstes benötigt werden, werden durch das `bindingTemplate` beschrieben und als Green Pages bezeichnet. Technische Details in dieser Verzeichniskategorie werden dabei durch `tModel`-Subeinträge repräsentiert. Diese stellen jedoch häufig nur einfache Verweise auf externe Daten dar, wie zum Beispiel einen Link auf das WSDL-Dokument eines Web Services. [OAS04, Cer02]

Die Struktur des verteilten UDDI-Verzeichnisses besteht aus verbundenen Verzeichnissen (*Affiliated Registries*), die zwar individuell ein vollständiges UDDI-Verzeichnis darstellen, aber ihre Daten untereinander austauschen und synchronisieren. Die einzelnen Registries selbst können wiederum aus einzelnen Knoten (*Nodes*) aufgebaut sein, die jeweils eine oder mehrere der UDDI-Funktionen auf den ihnen verfügbaren Daten ausführen. Über die eigentliche Implementierung dieser Knoten wird jedoch keine Angabe gemacht, sodass sich unterschiedliche Umsetzungen des UDDI-Standards entwickelt haben. [OAS04]

4.1.3. Common Object Request Broker Architecture

Auch wenn Web-Service-Architekturen eine verbreitete und explizite Realisierung einer dienstorientierten Architektur sind, so existieren andere Konzepte, die eine ähnliche Problematik adressieren und genutzt werden können, um das SOA-Modell umzusetzen. Hierzu gehört zum Beispiel die von der *Object Management Group* standardisierte Middleware-Plattform CORBA (*Common Object Request Broker Architecture*) zur Realisierung eines verteilten Objektsystems. Auch wenn Objekte aus der Sicht des Software-Engineerings nicht unbedingt eine vollständige Business-Funktion darstellen müssen, können sie dennoch dazu verwendet werden, Dienste zu realisieren. [DJMZ05, Ham05]

Die vom SOA-Modell geforderten Rollen des Dienstanbieters, -klienten und -verzeichnisses sind auch in der CORBA-Architektur vorhanden. Des

Weiteren muss der Nutzer eines Objektes ebenfalls nichts über die interne Struktur der Implementierung wissen, um es anzusprechen – somit bietet auch CORBA eine (gewisse) lose Kopplung. Zudem verbindet die CORBA-Middleware unterschiedliche Systeme und Programmiersprachen, sodass Interoperabilität und Technologieneutralität entsteht. Da die Lokalisierung von entfernten Objekten durch das CORBA-System vorgenommen wird und es für die Kommunikationspartner verborgen bleibt, wo sie sich befinden, ist auch die Forderung nach Orts- und Verteilungstransparenz gegeben. Somit wird die Common Object Request Broker Architecture durchaus einer allgemeinen dienstorientierten Architektur gerecht. [Ham05]

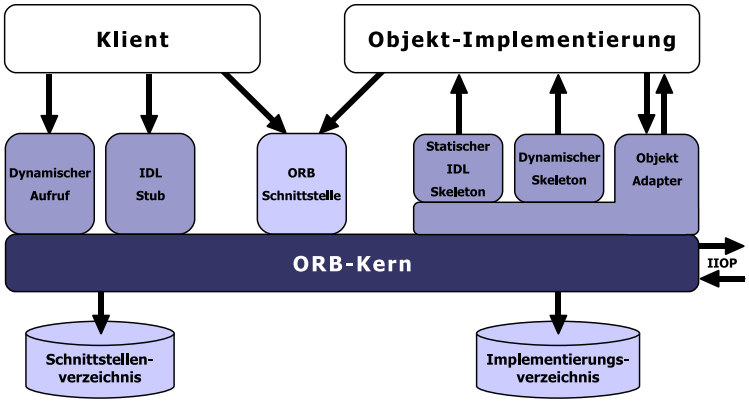


Abbildung 4.6.: Übersicht der CORBA-Komponenten (nach (Ham05, OMG99))

Der CORBA-Standard definiert die Architektur einer Middleware-Plattform zum Erzeugen und Nutzen von verteilten Objekten, jedoch nicht deren konkrete Implementierung. Das Kernstück der Architektur ist der *Object Request Broker* (ORB), über den die anderen Komponenten der Architektur miteinander verbunden sind (vgl. Abbildung 4.6). Er ermöglicht dabei die Kommunikation zwischen Klienten und entfernten Objekt-Implementierungen, während er dabei alle Aspekte verbirgt, die sich durch die Verteilung und Heterogenität des Systems ergeben. Neben dem Kernsystem definiert das Referenzmodell auch eine Reihe von Basisdiensten (*CORBAservices*) und spezifischen Diensten für einzelne Anwendungsdomänen (*horizontale und vertikale CORBAfacilities*), die im Folgenden jedoch nur dann beschrieben werden, wenn sie einen Teil des SOA-Modells abbilden. [Ham05, TS02]

ORB – Object Request Broker Der *Object Request Broker* bildet das Zentrum eines jeden verteilten CORBA-Systems. Seine Hauptaufgabe ist die

Vermittlung von (Dienst-)Aufrufen und den dazugehörigen Antworten zwischen zwei beliebigen Objekten. Um dies zu erreichen, muss der ORB zunächst die Objekt-Implementierungen, an die sich der Klient wendet, finden, diese dann auf den Eingang des Aufrufs vorbereiten und die eigentlichen Daten des Aufrufes an das Server-Objekt übertragen. Besonders wichtig ist hierbei die eindeutige Identifizierung und Adressierung der Klient- und Serverobjekte, die er mit Hilfe von Objektreferenzen verwirklicht. Zum Austausch dieser CORBA-Objektreferenzen auch über die Grenze des lokalen ORBs hinaus, verwenden CORBA-Systeme das als *Interoperable Object Reference* (IOR) bezeichnet Adressierungsschema, das sowohl den Kommunikationsendpunkt als auch das Objekt selbst eindeutig referenziert. [Ham05, TS02, Mah01]

Sind Klient und Server nicht durch die selbe ORB-Implementierung miteinander verbunden, kommunizieren die beiden lokalen ORBs mit Hilfe eines Standardprotokollschemas, dem *General Inter-Orb Protocol* (GIOP), das eine einheitliche Kommunikation auf Basis eines zuverlässigen und verbindungsorientierten Transportprotokolls spezifiziert. Die wohl verbreitetste Realisierung des GIOP ist das auf dem TCP/IP-Protokollstapel aufbauende *Internet-Inter-Orb-Protocol* (IIOP). [Ham05, TS02]

Da CORBA-Systeme unterschiedliche Systemplattformen und Programmiersprachen transparent miteinander verbinden, können Klient und Server nicht unbedingt direkt miteinander kommunizieren. Deshalb kapseln Stellvertreterobjekte die jeweiligen lokalen Aufrufe und Rückgaben sowie die Kommunikation zwischen den Objekten über den ORB. [Ham05]

Stellvertreterobjekte Entfernte Objekte kommunizieren in CORBA-Systemen nicht direkt miteinander, sondern über Stellvertreterobjekte, die eine lokale und programmiersprachenabhängige Repräsentation des entfernt zugreifbaren Objektes darstellen und den Zugriff auf den ORB kapseln. Zu ihren Kernaufgaben gehört vor allem das *Marshalling* und *Unmarshalling* der zu übertragenden Daten sowie der Umgang mit den transportspezifischen Details der Aufrufe. Dabei wird durch das Marshalling die systemabhängige Repräsentation der Daten in ein allgemeines Übertragungsformat gebracht und dann auf Empfängerseite durch das Unmarshalling wieder in das dort gültige Format konvertiert. Hierdurch ist es auch Systemen mit sehr unterschiedlichen Repräsentationsformen der Daten möglich, miteinander zu kommunizieren. [Ham05, Pop98]

Das Stellvertreterobjekt auf Seite des Klienten wird als *Stub* bezeichnet und stellt die Implementierung der Objektreferenz des entfernten Objekts

dar. Es kann durch den Klienten so angesprochen werden, als wenn sich das entfernte Objekt im lokalen Adressraum des Klienten befinden würde. Das Gegenstück auf der Implementierungsseite wird hingegen als *Skeleton* bezeichnet. Dieses stellt zunächst nur ein Objektskelett bereit, bei dem vordefinierte Methoden zum Marshalling und Unmarshalling vorhanden sind, während die funktionellen Methoden des entfernten Objektes jedoch von dem Programmierer selbst implementiert werden müssen. CORBA-Systeme unterstützen dabei sowohl statische als auch dynamische Aufrufe, was dazu führt, dass es auch statische und dynamische Stubs und Skeletons gibt. Dabei sind die dynamischen Stubs und Skeletons aus der dienstorientierten Sichtweise besonders wichtig, da mit ihnen ein dynamisches Finden und Einbinden von a priori unbekannten Dienstobjekten möglich wird. [Ham05, Pop98]

Die Schnittstelle der Stellvertreterobjekte wird dabei zunächst durch die allgemeine *Interface Definition Language* beschrieben und auf den jeweiligen Zielsystemen erst in die programmiersprachenabhängigen Stubs und Skeletons überführt. Um auch eine dynamische Suche entfernter Objekte zu ermöglichen, können ihre Schnittstellen im *Interface Repository* veröffentlicht werden.

IDL – Interface Definition Language Die *Interface Definition Language* ist neben dem ORB eine der grundlegenden Standards in der Common Object Request Broker Architecture. Sie stellt eine deklarative Sprache zur Beschreibung von Schnittstellen dar, mit der die Signaturen der Methoden eines Objektes und die benötigten Datentypen definiert werden können. Die IDL stellt dabei keine Sprachkonstrukte für die Beschreibung von Semantik oder Ablaufstrukturen bereit. IDL-Compiler überführen dann diese allgemeine Schnittstellenbeschreibung in die programmiersprachenabhängigen Stellvertreterobjekte (Stubs und Skeletons). [Ham05, TS02, Pop98]

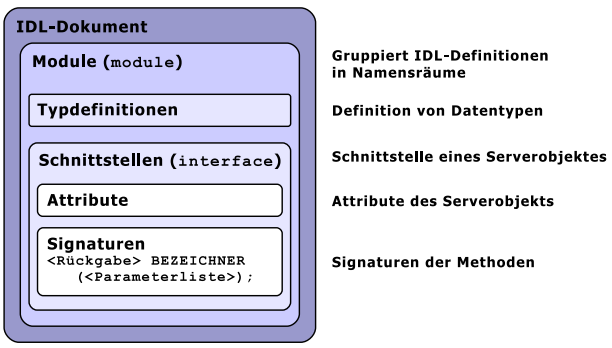


Abbildung 4.7.: Struktur eines einfachen IDL-Dokuments

Die Struktur eines einfachen IDL-Dokuments (vgl. Abbildung 4.7) gliedert sich in ein oder mehrere Module, die das Dokument in einzelne Namensräume strukturieren. Hiermit werden sowohl zusammengehörende Elemente gekapselt als auch ungewollte Namenskonflikte vermieden. Neben den von CORBA vorgegebenen Basisdatentypen, können innerhalb der Module auch eigene Datentypen definiert werden. Diese können entweder Basisdatentypen umdeklarieren oder komplexe Datentypen enthalten, die mit Hilfe von strukturierten Typen erzeugt werden. Die einzelnen Objekte eines Moduls werden dann durch ihre Schnittstellen beschrieben. Hierbei können neben den Methodensignaturen auch Attribute angegeben werden, die am Objekt von außen zugreifbar sein sollen. [Ham05, Pop98]

Interface Repository Das Schnittstellenverzeichnis (*Interface Repository*) ist einer der internen Dienste eines ORBs und stellt ein Verzeichnis über voll qualifizierte IDL-Schnittstellen dar. Das Verzeichnis dient dazu, zur Laufzeit eines Programms auf Schnittstellenbeschreibungen zuzugreifen, um beispielsweise dynamisch Objekte binden und ansprechen zu können. Dazu werden aus den Schnittstellenbeschreibungen und den im Verzeichnis enthaltenen Metainformationen zur Laufzeit Stubs für den Aufruf entfernter Objekte generiert. [Ham05, KHW+02, Pop98]

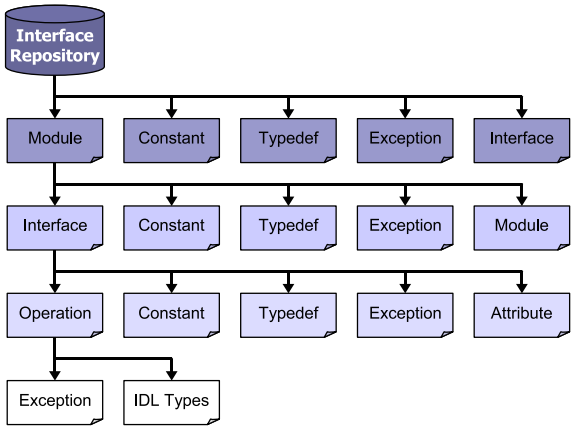


Abbildung 4.8.: Informationshierarchie des Information Repository (nach (Pop98))

Das Interface Repository ist ebenfalls mittels einer IDL-Schnittstelle beschrieben und über den ORB als entferntes Objekt ansprechbar. Dabei sind die enthaltenen Informationen hierarchisch geordnet (vgl. Abbildung 4.8) und werden in Form von entfernten Objekten, die vom Interface-Repository-Objekt abgeleitet sind, den Klienten zur Verfügung gestellt. Dabei arbeitet das Verzeichnis eng mit dem Klienten zum dynamischen Auf-

ruf entfernter Dienste zusammen. Dieser kann in den Informationen suchen, die für ihn notwendigen Informationen auslesen und so dynamische Aufrufe an einer Schnittstelle erzeugen. [Ham05, Pop98]

Objekt Adapter Um allgemeine Funktionen zu zentralisieren, die vielen Objekt-Implementierungen gemein sind, sieht die CORBA-Architektur *Objekt Adapter* vor. Dabei verwalten diese Adapter Objekt-Implementierungen in ihren Lebensphasen. Zu diesen Phasen gehören zum Beispiel das Initialisieren und Löschen sowie das Passivieren und Aktivieren der Objekte. Zudem ist der Objekt Adapter für das Generieren der eindeutigen Objektreferenzen verantwortlich. [Ham05, Pop98]

Implementation Repository Um den Objekt Adapter bei der Verwaltung von Objekt-Implementierungen zu unterstützen, steht dem Kernsystem ein Implementierungsverzeichnis (*Implementation Repository*) zur Verfügung. Dabei enthält das Verzeichnis unter anderem Informationen zum lokalen Pfad der Objekt-Implementierung, zum Initialisieren des Objektes und zu den Zugriffsrechten. [Ham05, Pop98]

4.1.4. Vergleich von CORBA und Web Services

Die vorherigen Abschnitte haben gezeigt, dass sowohl mit Web Services als auch mit CORBA-Systemen unterschiedliche Umsetzungen einer SOA vorhanden sind. Beide vorgestellten Ansätze haben dabei ihre eigenen Lösungen und Standards für die wichtigsten Aufgaben in einer dienstorientierten Architektur entwickelt (vgl. Tabelle 4.1).

So kommunizieren Klienten und Dienstanbieter in einer Web-Service-Architektur über SOAP-Nachrichten, die mit Hilfe des Hypertext-Transfer-Protokolls (*HTTP*) übermittelt werden. CORBA-Systeme hingegen tauschen ihre Informationen über das vom allgemeinen GIOP abgeleitete Internet-Inter-ORB-Protokoll (*IIOP*) aus. Dabei werden die Kommunikationsendpunkte jeweils durch technologietypische Referenzen, nämlich einer URL für einen Web Service und einer IOR für CORBA-Objekte, definiert.

Auch die Beschreibung der Schnittstelle ist in jedem der Ansätze einzeln gelöst worden. Dabei beschreibt ein IDL-Dokument eine hierarchische

	CORBA	Web Services
Nachrichtenaustausch	IIOP, GIOP	SOAP, HTTP
Endpunktreferenz	IOR	URL
Schnittstellenbeschreibung	IDL	WSDL
Naming, Verzeichnis	Naming Service, Interface Repository, Trader Service	UDDI
Zustand	zustandsbehaftet	zustandslos
Kodierung der Nutzdaten	CDR (binär)	XML-basiert

Tabelle 4.1.: Gegenüberstellung von CORBA und Web Services (nach (DJMZ05))

Sicht auf die Schnittstelle eines CORBA-Objektes, wogegen ein WSDL-Dokument eine nachrichtenorientierte Beschreibung des Dienstzugriffs anbietet. [ACS06]

Im Bereich von Naming- und Verzeichnisdiensten bieten Web Services mit UDDI einen integrierten Ansatz, wogegen ein CORBA-System unterschiedliche Dienste hierzu bereithält. Dies sind vor allem der Naming Service, das Interface Repository und der Trader Service.

Ein grundlegender und konzeptioneller Unterschied zwischen Web Services und CORBA-Systemen liegt in dem von ihnen verwendeten Zustandskonzept: So sind Web Services per se als zustandslos definiert, wogegen CORBA-Objekte durchaus einen Zustand über mehrere Aufrufe eines Klienten besitzen können – aber nicht müssen.

Zudem unterscheiden sich die beiden Ansätze auch in der Art, wie sie die übermittelten Daten kodieren. Während im CORBA-Ansatz die Daten mit Hilfe der *Common Data Representation* (CDR) binär repräsentiert werden, überträgt ein Web Service seine Daten XML-basiert und somit textuell beschrieben.

Trotz dieser vordergründigen starken technologischen Unterschiede können CORBA-Systeme und Web Services aufgrund des gemeinsamen dienstorientierten Ansatzes durchaus untereinander kooperieren und voneinander profitieren. CORBA kann sogar als eine Möglichkeit gesehen werden, einen Web Service zu implementieren. Um dabei beide Welten miteinander zu vereinen, werden vermittelnde Komponenten – auch als *Bridges* bezeichnet – eingesetzt (vgl. Abbildung 4.9).

Bridges sind dabei sowohl SOAP-Server als auch Klient eines entfernten CORBA-Objekts. Sie können entweder statisch ein Objekt kapseln

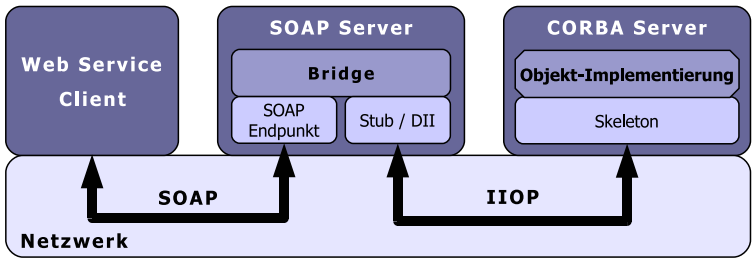


Abbildung 4.9.: Anbindung von Web-Service-Klienten an CORBA-Implementierungen

oder Aufrufe erst zur Laufzeit an ein Objekt binden und so eine dynamische Kapselung vornehmen. Hierzu muss zunächst eine Konvertierung von der IDL- auf eine WSDL-Beschreibung der Objektschnittstelle vorgenommen werden, um diese den Web-Service-Klienten zugänglich machen zu können. Wie eine solche Abbildung zwischen IDL und WSDL aussehen kann, ist dabei ebenfalls durch den CORBA-Standard definiert (siehe [Obj06]). Danach ist es die Aufgabe der Bridge, Web-Service-Aufrufe in CORBA-Aufrufe umzusetzen. Hierzu müssen die empfangenen Nachrichten auf die entsprechenden Methodenaufrufe abgebildet werden. Ergebnisse werden dann umgekehrt auch wieder als Web-Service-Nachrichten an den Klienten zurückgesendet. [ACS06, HZJZ04]

4.2. Prozesse zur Komposition komplexer Dienste

Einzelne Dienste können durch Kompositionsmechanismen zu komplexen Diensten aggregiert werden. Dabei können in dienstorientierten Umgebungen die Strukturen solcher nicht-atomarer, komplexer Aufgaben und damit der abstrakten Sicht auf die dahinter stehenden Funktionalitäten durch Prozesse beschrieben werden:

> *A **process** consists of a number of tasks that need to be carried out and a set of conditions that determine the order of the tasks.*
>
> [AH02, Seite 4]

Ein Prozess beschreibt also die Struktur einer Aufgabe einerseits durch die enthaltenen Tätigkeiten (*Tasks*) und andererseits durch die Abfolge, in der diese angeordnet sind. Dabei bildet jede Tätigkeit eine logische Einheit, die entweder eine atomare Funktionalität darstellt oder selbst wieder als Prozess definiert sein kann. Ausgeführt werden die Tätigkeiten jeweils durch eine Ressource, die nicht unbedingt ein rechnergestütztes System

sein muss, sondern auch Personen oder Organisationen umfassen kann. [AH02]

Wie Prozesse aufgebaut sind und wie sie auf (mobile) Systeme abgebildet werden können, wird in den folgenden Abschnitten gezeigt. Hierbei stehen Geschäftsprozesse im Vordergrund der Betrachtung, da sie neben der Struktur auch das Ergebnis eines Prozesses berücksichtigen. Geschäftsprozesse besitzen also immer ein Ziel, das erreicht werden soll, und sind deshalb als Beschreibung komplexer Aufgaben in mobilen Szenarien besonders geeignet. [Rei03]

Aufgabe und Definition eines Geschäftsprozesses

Geschäftsprozesse (*Business Processes*) sind zielorientierte Prozesse, die mit einem bestimmten (wirtschaftlichen) Nutzen verbunden sind. Sie können damit als eine Untermenge allgemeiner Prozesse betrachtet und wie folgt definiert werden:

> ***Business Process*** *– The definition of the tasks and the sequence of those tasks necessary to deliver a business function.*
>
> [Dav05, Seite 2]

Dabei stellt das abstrakte Modell eines Geschäftsprozess eine konzeptionelle Sicht auf eine Aufgabe dar, welche die allgemeine Organisation von Ressourcen und Diensten beschreibt, durch die ein bestimmtes Ergebnis – zum Beispiel ein Produkt oder eine Dienstleitung – erreicht werden kann. Das Modell repräsentiert damit also eine Schablone (*Template*), mit deren Hilfe konkrete Instanzen eines Geschäftsprozesses erzeugt werden können. Werden dabei einzelne Abschnitte bis hin zum gesamten Prozess rechnergestützt ausgeführt, werden diese Teile des Geschäftsprozesses als *Workflow* bezeichnet (vgl. Abbildung 4.10). [Rei03, LR00]

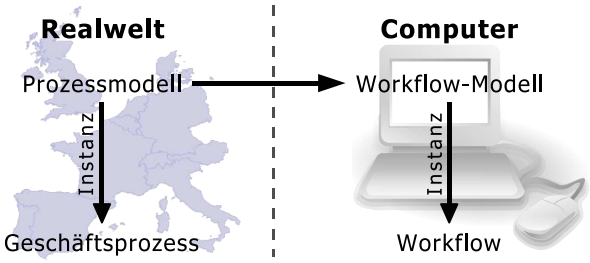


Abbildung 4.10.: Beziehung zwischen Prozessen und Workflows (nach (LR00))

Workflows sind also Geschäftsprozesse, die mit Hilfe von Computern ablaufen. Dabei können sie entweder durch Prozessbeschreibungssprachen

und Universalverwaltungssystemen ausgeführt werden oder aber sie werden jeweils als eigene Spezialanwendung umgesetzt. [LR00]

4.3. Modellierung von Geschäftsprozessen als Workflows

Allgemein können bei der Prozessmodellierung verschiedene Aspekte eines Geschäftsprozesses abgebildet werden. Es handelt sich also immer um eine vereinfachte Abbildung eines Ausschnitts des betrachteten Systems. Deshalb muss zu Beginn geklärt werden, welche Aspekte in welcher Art modelliert werden sollen und wie die benötigten Informationen möglichst optimal abgebildet werden können. Die Dokumentation einer solchen Modellierung kann hierbei prinzipiell auf vielfältige Weise vorgenommen werden: So können die Prozesse auf einfache Weise als beschreibender Text, mit Hilfe einer tabellarischen Darstellung oder grafisch beschrieben werden, wobei jedoch eine solche eher informelle Beschreibung meist keine einheitliche Interpretation oder automatisierte Verarbeitung der Daten ermöglicht. Deshalb sollte eine uniforme und standardisierte Notation gewählt werden, sodass gleichartige Modelle entstehen und damit ein einheitliches Verständnis ermöglicht wird. Dies gilt vor allem, wenn der gesamte Prozess oder Teile von ihm als Workflow, also computergestützt, ablaufen sollen. [All05]

Eine zentrale Rolle bei der Modellierung von Geschäftsprozessen nimmt meist die Abbildung der im Prozess durchzuführenden Aufgaben und deren zeitlich-logische Abfolge (*Kontrollfluss*) ein. Deshalb ist ein Workflow auch durch die Menge seiner Arbeitsschritte (*Aktivitäten*) mit einer zugehörigen partiellen Ordnung definiert, mit deren Hilfe ein vordefiniertes Ziel erreicht werden soll. Dabei fokussiert der Workflow auf die Struktur des übergeordneten Arbeitsablaufs und nicht auf den individuellen Arbeitsschritt. Das Ergebnis der Modellierung ist dann ein *Workflow-Modell* oder *-Schema*, das einen bestimmten Prozesstyp beschreibt und als Vorlage für die Ausführung einzelner *Workflow-Instanzen* dient (vgl. Abbildung 4.10). Es sind also im Workflow-Modell alle möglichen Pfade durch den Geschäftsprozess und die Regeln, die den konkreten Pfad bestimmen, beschrieben. Bei der Ausführung werden dann die Regeln auf die verfügbaren Daten angewendet und so ein konkreter Prozessablauf erzeugt. [ZK07, All05, DAH05, Ell99]

4.3.1. Perspektiven eines Workflows

Die Modellierung von Geschäftsprozessen als Workflow ist eine Voraussetzung zur rechnergestützten Ausführung einzelner Prozessinstanzen. Dabei können Workflows unterschiedliche Perspektiven abbilden, die im Folgenden kurz beschrieben werden:

Ressourcen und Ressourcen-Management Für jeden Workflow muss festgelegt und dokumentiert werden, welche Ressourcen in welcher Anzahl zu seiner Ausführung benötigt werden. Dabei bezeichnet eine Ressource alle Arten von Objekten, die im Prozess zur Ausführung einer Aktivität benötigt werden. Hierzu werden bei der Modellierung im Allgemeinen lediglich Ressourcenklassen definiert, denen dann zur Laufzeit konkrete Instanzen der Ressource zugeordnet werden. [All05]

Organisationseinheiten Um die Verflechtung und Verantwortlichkeiten von betrieblichen Organisationseinheiten bei der Bearbeitung eines Geschäftsprozesses aufzeigen und dokumentieren zu können, kann es sinnvoll sein, die Organisationseinheiten auch im Workflow zu modellieren. Dabei können die Einheiten – wie beispielsweise Abteilungen, Gruppen oder Rollen des Unternehmens – als spezielle Ressourcen angesehen werden, die neben den Beziehungen zwischen den Einheiten auch Attribute wie Kompetenzen oder Verantwortlichkeiten definieren können. [All05]

Aufgaben und Aufgaben-Management Die Sicht auf die im Geschäftsprozess enthaltenen Aktivitäten und deren zeitlich-logische Abfolge ist eine der Kernsichten in einem Workflow-Modell. Dieses auch als Kontrollfluss bezeichnete Teilmodell eines Workflows (vgl. Abschnitt 4.3.2) bezieht seine Relevanz vor allem daraus, dass sich andere Sichten meist auf die modellierten Aufgaben des Prozesses beziehen. [All05]

Daten und Datenfluss Um einen Workflow automatisiert ausführen zu können, müssen sowohl die von den Aktivitäten benötigten und erzeugten Daten (Datenfluss) als auch Metadaten des Workflows selbst modelliert werden (vgl. Abschnitt 4.3.3). Aber auch historische und administrative Daten, die zum Beispiel Angaben über den Zustand des Prozesses machen, können bei der Beschreibung berücksichtigt werden. [All05]

Zeitliche Aspekte Besonders im Hinblick auf die Garantie von Qualitätsmerkmalen sind auch zeitliche Aspekte innerhalb eines Geschäftsprozesses interessant. Dabei beziehen sich diese Informationen meist auf Fristen zur Ausführung von Aktivitäten oder deren Dauer. Aber auch die zeitliche Verfügbarkeit von Ressourcen und andere zeitliche Einschränkungen, die im Geschäftsprozess berücksichtigt werden müssen, sollten modellierbar sein. [All05]

Anwendungen Die Sicht auf die zur Ausführung benötigten Anwendungen fokussiert auf die tatsächlich verwendeten Programme. Dabei stehen die Anforderungen im Mittelpunkt, die beim Aufruf, bei der Ausführung und in Fehlersituationen auftreten. [All05]

Geschäftsregeln Durch Geschäftsregeln werden Richtlinien oder Strategien eines Unternehmens beschrieben. Diese übergeordneten und meist nicht-funktionalen Leitlinien bilden dann einen Rahmen zur Ausführung mehrerer Geschäftsprozesse und sollten deshalb nicht im einzelnen Workflow modelliert werden. Aus diesem Grund werden sie oftmals in einem eigenen Modell beschrieben – zum Beispiel in Form einer Sammlung von ECA-Regeln (*Event-Condition-Action*). [All05]

Behandlung von Ausnahmen Besonders wichtig kann die Berücksichtigung von Fehler- beziehungsweise Ausnahmesituationen bei der Modellierung eines Geschäftsprozesses sein. Dies gilt besonders dann, wenn äußere und somit nicht von der Prozessausführung kontrollierbare Einflüsse oder ein unzuverlässiges Ausführungsumfeld – wie zum Beispiel in mobilen Umgebungen – bestehen. In solchen Situationen kann es sinnvoll sein, Prozessbeteiligte bei der Behebung einer Ausnahmesituation durch modellierte Fehlerbehandlungsmaßnahmen zu unterstützen. [All05]

Interorganisationelle Kooperation Immer dann, wenn ein Geschäftsprozess über Organisationsgrenzen hinweg verläuft, kann es sinnvoll sein, die Zusammenarbeit zwischen den beteiligten Institutionen mit abzubilden. Dabei konzentriert sich die Sicht auf die interorganisationelle Kooperation auf die Schnittstellen zwischen den Unternehmen, die auszutauschenden Datenformate, die Aufteilung der Workflow-Teilmodelle und die Regeln zur Zuweisung einzelner Aufgaben. [All05]

4.3.2. Abbildung des Kontrollflusses

Die Abbildung der in einem Geschäftsprozess enthaltenen Aktivitäten und deren Interaktion werden in einem Workflow-Modell durch den Kontrollfluss definiert. Eine solche Ausführungsreihenfolge der Aufgaben kann durch unterschiedliche syntaktische Strukturen, wie zum Beispiel *Block- oder Graphstrukturen*, beschrieben werden. Dabei wird in blockstrukturierten Modellen – ähnlich wie in einer Programmiersprache – eine Schachtelung der Prozesselementen mit Hilfe von Angaben zur Bildung von Sequenzen, bedingten Ausführungspfaden und Iterationen erreicht. Graphbasierte Modelle hingegen beschreiben den Kontrollfluss des Prozesses mit Hilfe von Knoten und Transitionen. Dabei repräsentieren die Knoten des Graphen die einzelnen Aktivitäten und die Transitionen die Übergänge zwischen ihnen. [ZK07]

4.3.2.1. Aktivität

Eine Aktivität (*Activity*) ist eine abstrakte Beschreibung einer der Arbeitsschritte innerhalb eines Prozesses (N), die eine Menge von Eingabedaten ($i(A)$) in eine Menge von Ausgabedaten ($o(A)$) überführt. Eine Aktivität stellt somit formal eine Operation dar und kann dementsprechend als Funktion definiert werden: $A \in N : i(A) \rightarrow o(A)$. Die Beschreibung der Aktivität enthält dabei neben Angaben zu den Ein- und Ausgabedaten auch Informationen über die Art der zugrunde liegenden Tätigkeit und zu den Akteuren, die an ihrer Ausführung beteiligt sind. [LR00]

Aktivitäten sind entweder elementar (*Elementary Activity*) oder sie kapseln als zusammengesetzte Aktivität (*Compound Activity*) einen Subprozess. Elementare Aktivitäten stellen dabei eine grundlegende Tätigkeit dar, die nur aus einem primitiven Arbeitschritt besteht und von einem einzelnen Akteur ausgeführt werden kann. Dabei können Aktivitäten entweder automatisch, manuell oder durch die Interaktion von Rechnern und Personen ausgeführt werden. [LR00, Ell99]

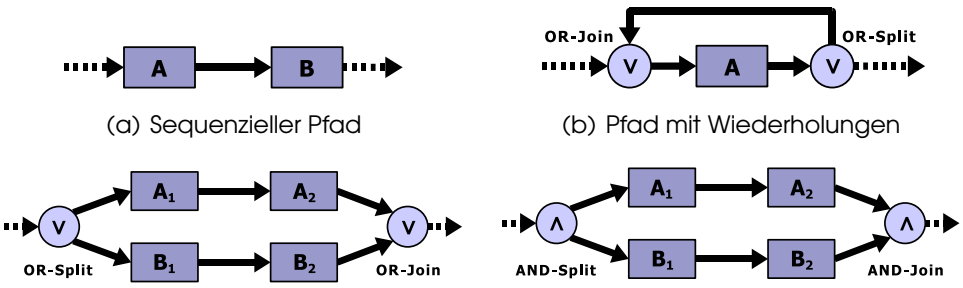
4.3.2.2. Intraprozesskoordination

In graphstrukturierten Workflows werden die Übergänge zwischen den einzelnen Aktivitäten durch *Kontrollflusskonnektoren* beschrieben. Dabei gibt ein solcher Konnektor an, welche Aktivitäten unter welcher Bedingung miteinander verknüpft sind. Die Kontrollflusskonnektoren (E) können also als kartesisches Produkt zwischen den Aktivitäten (N) und einer

Menge von Übergangsbedingungen ($\mathcal{C}$) definiert werden: $E \subseteq N \times N \times \mathcal{C}$. [LR00]

Kontrollflusskonnektoren stellen gerichtete Transitionen dar und besitzen somit für eine Aktivität entweder aus- oder eingehenden Charakter. Dabei kann eine Aktivität auch mehrere eingehende und ausgehende Konnektoren besitzen, die dann spezifizieren, ob die Aufgaben im Prozess sequenziell oder parallel ausgeführt werden können. Der Start eines Prozesses ist durch die Aktivitäten ohne eingehenden Konnektoren definiert. Aktivitäten ohne ausgehende Konnektoren hingegen bezeichnen den Abschluss des Workflows. [ZK07, LR00]

Zur Abbildung der Ausführungsfolge von Aktivitäten stehen dem Modellierer eine Vielzahl unterschiedlicher so genannter *Kontrollflussmuster* zur Verfügung. Zu den elementaren Mustern gehören dabei neben der Sequenz von Aktivitäten auch die Iteration, die Auswahl und die Parallelausführung (vgl. Abbildung 4.11). [DAH05]



(a) Sequenzieller Pfad (b) Pfad mit Wiederholungen

(c) Alternative Pfade (d) Parallele Pfade

Abbildung 4.11.: Elementare Kontrollflussmuster (z. T. nach (DAH05))

Sequenz Bei einer Sequenz werden die Aktivitäten einer Ausführungsfolge eine nach der anderen abgearbeitet. Sind also zwei Aktivitäten durch einen Kontrollflusskonnektor verbunden, kann die zweite Aktivität (B) erst dann begonnen werden, wenn die Ausführung der vorhergehenden (A) beendet ist (vgl. Abbildung 4.11(a)). Dabei kann sich die Reihenfolge, zum Beispiel durch Datenabhängigkeiten, aus der zu modellierenden Struktur des Geschäftsprozesses ergeben oder sie wird durch den Modellierer festgelegt. [ZK07, DAH05]

Im einfachsten Fall ist die sequenzielle Ausführung zweier Aktivitäten unbedingt. Hierbei beginnt die zweite Aktivität direkt nach Abschluss der ersten und es ist durch die Transition lediglich die Ausführungsreihenfolge festgelegt. Durch die Angabe zusätzlicher *Transitionsbedingungen* sowie

Aktivierungs- und *Ausgangsbedingungen* kann jedoch von der unmittelbaren Ausführung abgewichen werden. Bei der Transitionsbedingung wird vor dem Übergang zur nächsten Aktivität die angegebene Bedingung geprüft und die Transition bei einer negativen Evaluierung nicht ausgeführt. Aktivierungs- und Ausgangsbedingungen verhalten sich ähnlich, sind aber direkt einer Aktivität zugeordnet und nicht einem einzelnen Kontrollflusskonnektor. Sie sind also unabhängig von den Pfaden, in denen die Aktivität vorkommt. Die durch die Aktivierungsbedingung beschriebene Voraussetzung entscheidet hierbei, ob eine Aktivität gestartet werden kann und die Ausgangsbedingung, ob die Aktivität erfolgreich beendet wurde. Dabei dienen Transitions- sowie Aktivierungs- und Ausgangsbedingungen oftmals dazu, nicht-funktionale Aspekte – wie zum Beispiel Qualitätsmerkmale –, die einen Rahmen für die Ausführung eines Geschäftsprozesses definieren, im Workflows zu berücksichtigen (vgl. Abschnitt 4.3.6). [ZK07, LR00]

Auswahl Um alternative Ausführungspfade in einem Workflow zu beschreiben, können bei der Modellierung Konstrukte zur Auswahl der Nachfolgeaktivität (*OR-Split*) eingefügt werden (vgl. Abbildung 4.11(c)). Dabei wird aus der Menge aller möglichen Nachfolger einer erfolgreich beendeten Aktivität eine Untermenge bestimmt, deren Elemente als nächstes ausgeführt werden sollen. Die Stellen, an denen sich alternative Pfade wieder vereinigen, werden analog als *OR-Join* bezeichnet. [DAH05]

Die Voraussetzungen, unter denen ein Pfad in die Menge der möglichen Nachfolger aufgenommen wird, kann dabei wieder durch Transitionsbedingungen beschrieben werden. Dabei werden nur die Aktivitäten berücksichtigt, deren Transitionsbedingung positiv evaluiert werden kann. Wie viele alternative Prozesspfade entstehen, hängt von der Art des verwendeten OR-Splits ab. Allgemein wird jedoch ein exklusiver OR-Split verwendet, der nicht-deterministisch eine Aktivität aus der Untermenge aller möglichen Nachfolger auswählt und so nur einen der alternativen Pfade verfolgt. [KHA03]

Parallelausführung Besonders im Hinblick auf eine mögliche Beschleunigung der Prozessausführung können auch parallele Ausführungspfade definiert werden. Hierbei werden mindestens zwei oder aber mehrere Aktivitäten unabhängig voneinander ausgeführt (vgl. Abbildung 4.11(d)). Das Aufteilen des Kontrollflusses in unabhängige Pfade wird dabei durch *AND-Splits* modelliert. Durch *AND-Joins* werden parallele Ausführungspfade dann wieder in einem Punkt synchronisiert. Durch die Angabe von Join-

Bedingungen, kann hierbei die Synchronisation und so der Fortgang des Workflows beliebig gesteuert werden. Hierdurch kann zum Beispiel eine Zwei-aus-Drei-Entscheidung modelliert werden. [DAH05, LR00]

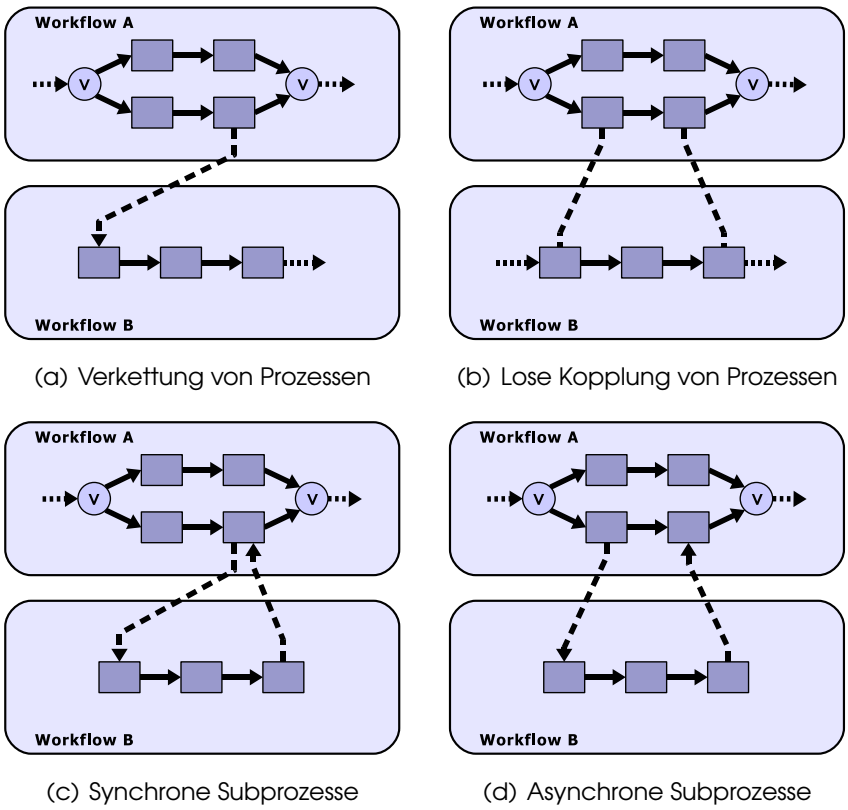
Iteration Iterationen dienen dazu, einzelne Aktivitäten oder eine Folge von Aktivitäten mehrfach auszuführen. Dabei kann die Iteration entweder durch die Anzahl ihrer Durchläufe oder durch das Eintreten einer Endbedingung wieder verlassen werden. Zwar kann eine Iteration auch durch OR-Splits und OR-Joins modelliert werden (vgl. Abbildung 4.11(b)), da jedoch eine wiederholte Ausführung eine häufig benötigte Anweisung ist, gibt es oftmals eigene Konstrukte, um diese Kontrollflussschleifen zu beschreiben. Zudem kann durch die Verwendung von expliziten Schleifenanweisungen die Verständlichkeit der Modelle erhöht werden. [ZK07, DAH05]

4.3.2.3. Interprozesskoordination

Die Koordination zwischen Prozessen ist immer dann notwendig, wenn entweder Aktivitäten eines Prozesses durch Subprozesse definiert sind oder wenn Daten und Ereignisse fremder Prozesse den lokalen Prozess beeinflussen. Diese *Interoperabilität* zwischen den Prozessen ist dabei durch unterschiedliche Verflechtungs- und Verteilungsgrade gekennzeichnet. Hierbei gibt der Verflechtungsgrad an, wie stark die Ausführung der Prozesse voneinander abhängt (vgl. Abbildung 4.12). Der Verteilungsgrad hingegen beschreibt, inwieweit die Verflechtung der Prozesse auch über organisationelle Grenzen hinweg besteht.

Die Verteilung kann von einer nur logischen Entkopplung der Prozesse im lokalen System über die Distribution der Prozesse auf unterschiedliche Systeme einer Organisation bis hin zu interorganisationellen Verteilungsformen reichen. Je nach Verflechtungsgrad müssen dabei mehr oder weniger Details des lokalen Workflows nach außen sichtbar gemacht werden. Hierfür kann zum Beispiel der *Workflow-View-Ansatz* verwendet werden, der es ermöglicht, eine eingeschränkte Sicht auf den lokalen Prozess zu erzeugen und nach außen verfügbar zu machen. [SO04, MA99]

Verkettete Prozesse Bei der Verkettung von Prozessen (*Chained Processes*) führt die Ausführung einer Aktivität des lokalen Prozesses dazu, dass ein Subprozess autonom gestartet und ausgeführt wird. Der aufrufende Prozess wird – ohne weiteres Interesse an dem neu gestarteten Prozess



(a) Verkettung von Prozessen (b) Lose Kopplung von Prozessen

(c) Synchrone Subprozesse (d) Asynchrone Subprozesse

Abbildung 4.12.: Interoperabilität von Prozessen (nach (LR00, MA99))

– fortgeführt oder sogar terminiert (vgl. Abbildung 4.12(a)). Bis auf das Starten des Subprozesses bleiben also beide Prozesse autonom und weisen keine weitere Verflechtung auf. [LR00, MA99]

Lose gekoppelte Prozesse Lose gekoppelte Workflows stellen zunächst autonome Prozesse dar, die sich jedoch an vordefinierten Stellen untereinander durch den Austausch von Ereignisnachrichten synchronisieren, um die korrekte Ausführung des übergeordneten Gesamtgeschäftsprozesses zu sichern (vgl. Abbildung 4.12(b)). Um diese Synchronisation zu erreichen, werden so genannte *Check Points* in den Prozess eingefügt, an denen Ereignisnachrichten produziert beziehungsweise konsumiert werden. Diese Art der Interaktion kann zum Beispiel dazu eingesetzt werden, um zu überprüfen, ob ein anderer Prozess bereits terminiert ist. [Aal00, MA99]

Eingebettete synchrone Subprozesse Im Fall der eingebetteten synchronen Subprozesse startet eine Aktivität einen zweiten Prozess und wartet dann auf die Terminierung des Subprozesses, um dessen Ergebnisse lokal

weiterverarbeiten zu können (vgl. Abbildung 4.12(c) und Abschnitt 3.2). Dieses, auch als hierarchische Modell bezeichnete, Verflechtungsmuster führt so zu einer engen Kopplung und damit zur Synchronisation beider Prozesse. [LR00, MA99]

Eingebettete asynchrone Subprozesse Auch bei den eingebetteten asynchronen Subprozessen wird ein neuer Prozess durch eine Aktivität in einem Workflow gestartet. Jedoch laufen beide Prozesse zunächst autonom voneinander weiter, bis der aufrufende Prozess einen Synchronisationspunkt (*Rendezvous Point*) erreicht, an dem die Ergebnisse des Subprozesses konsumiert werden sollen (vgl. Abbildung 4.12(d)). Stehen die Ergebnisse zu diesem Zeitpunkt noch nicht fest, so wartet der aufrufende Prozess am Synchronisationspunkt bis der Subprozess terminiert. [MA99]

4.3.3. Beschreibung des Datenflusses

Daten sind ein wichtiger Teil eines Workflow-Modells, da sie die Ausführung einer Prozessinstanz maßgeblich beeinflussen, indem sie Ein- und Ausgabedaten der einzelnen Arbeitsschritte darstellen und auf ihrer Grundlage die Bedingungen im Kontrollfluss ausgewertet werden. Sämtliche Daten, die durch das Prozessmodell definiert und während der Ausführung einer Prozessinstanz verarbeitet werden, stellen die so genannten Prozessdaten dar und definieren den aktuellen Ausführungskontext der Prozessinstanz. Welche dieser Daten der einzelnen Aktivität oder Bedingung als Eingabe beziehungsweise welche Ausgabedaten welchem Datenelement zugeordnet werden, ist in einem Workflow-Modell durch das *Datenmodell* und den *Datenfluss* beschrieben. [LR00]

Daten, die als Eingabe einem Element des Prozesses zur Verfügung stehen, werden als dessen *Input-Container* bezeichnet und umfassen in der Regel sämtliche Daten des übergeordneten Geschäftsprozesses. Äquivalent werden die Daten, die durch eine Aktivität erzeugt oder verändert werden als dessen *Output-Container* deklariert. Zunächst sind die Container eines Elements nur lokal verfügbar und müssen erst durch die explizite Angabe von Übergangsregeln zwischen den Output- und Input-Containern zweier Elemente sowie eventuell benötigten Konvertierungsfunktionen verfügbar gemacht werden. Diese Übergänge werden dabei durch Datenkonnektoren spezifiziert, die in ihrer Gesamtheit den Datenfluss des Prozesses bestimmen (vgl. Abbildung 4.13). [LR00]

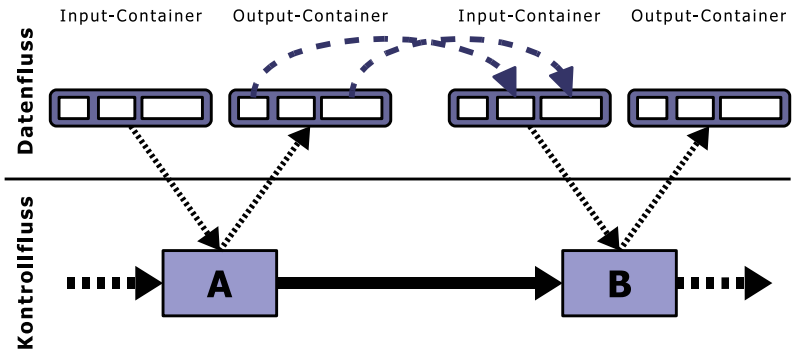


Abbildung 4.13.: Datenfluss zwischen Aktivitäten (nach (ZK07, LR00))

Jeder Input- und Output-Container stellt eine Datenstruktur dar, die die Domänen der enthaltenen Daten definiert. Dabei können vordefinierte Basisdatentypen verwendet werden oder komplexere Datentypen durch Strukturierungs- und Aggregationstypen konstruiert werden. Um diese Datenstrukturen wieder verwendbar zu machen, werden sie im Workflow-Modell separat von den Containern definiert. [LR00]

4.3.4. Prozessteilnehmer

Prozessteilnehmer (*Participants*) sind ein wichtiger Bestandteil von Prozessbeschreibungen. Teilnehmer können dabei Personen, Programme, Rollen oder andere zur Ausführung einer Aktivität benötigten Einheiten sein. Dabei stellt eine Rolle einen benannten Platzhalter für eine Gruppe von Teilnehmern dar, der erst zur Laufzeit in einen konkreten Teilnehmer aufgelöst wird. Dabei kann ein Teilnehmer durchaus mehrere Rollen innehaben. Durch die Verwendung von Rollen kann die Strategie der *späten Bindung* umgesetzt werden, die eine von konkreten Teilnehmern unabhängige und damit abstrakte Definition von Prozessen ermöglicht. [ZK07, Ell99]

Ist ein Teilnehmer eine Person – auch als *Actor* bezeichnet –, so kann sie entweder selbst eine manuelle Aktivität ausführen oder die Ausführung von automatischen Aktivitäten überwachen. Allen Teilnehmern gemein ist, dass sie in irgendeiner Weise an der Abarbeitung der ihnen zugeordneten Aktivitäten beteiligt sind. [Ell99]

4.3.5. Transaktionen

Um in einem Prozess Aktivitäten zu logischen Verarbeitungseinheiten zu verknüpfen, die nur dann korrekt ausgeführt sind, wenn alle enthaltenen Arbeitsschritte vollständig abgearbeitet wurden, sind Konzepte zur

Transaktionsverarbeitung notwendig. Jedoch sind die aus eng gekoppelten Systemen bekannten *klassischen (verteilten) Transaktionen*, welche die ACID-Eigenschaften erfüllen (vgl. Tabelle 4.2), für prozessorientierte und damit lose gekoppelte Systeme nicht immer ausreichend. Vielmehr werden erweiterte Konzepte (*Advanced Transaction Models*) benötigt, die die strikten Forderungen nach *Atomarität*, *Konsistenz*, *Isolation* und *Dauerhaftigkeit* zum Teil aufweichen. Dies beruht insbesondere darauf, dass prozessorientierte Umgebungen durch längere Ausführungszeiten, dem Einbeziehen von nicht-zurücksetzbaren Aktivitäten und vor allem durch die Heterogenität der beteiligten Systeme andere Voraussetzungen haben, als dies bei klassischen (verteilten) Transaktionssystemen der Fall ist. Deshalb müssen Transaktionen in dienst- und prozessorientierten Systemen eher lose gekoppelt sein und auch Aktivitäten einschließen können, die nur durch semantisches Zurücksetzen (*Kompensation*) rückgängig zu machen sind. [HW05, WY05, EN02]

Eigenschaft	Beschreibung
Atomarität	Alle Aktionen einer Transaktion stellen eine atomare Verarbeitungseinheit dar. Sie werden entweder alle erfolgreich oder aber überhaupt nicht ausgeführt.
Konsistenz	Transaktionen bewahren die Konsistenz eines Systems, da sie es von einem konsistenten Zustand in einen anderen konsistenten Zustand überführen.
Isolation	Gleichzeitig ausgeführte Transaktionen dürfen sich nicht gegenseitig beeinflussen. Das Ergebnis der Transaktion muss deshalb immer so sein, als wenn sie einzeln ausgeführt wurde.
Dauerhaftigkeit	Die durch eine Transaktion ausgeführten Änderungen müssen nach deren Beendigung dauerhaft sein und dürfen nicht verloren gehen.

Tabelle 4.2.: ACID-Eigenschaften (nach (EN02))

Klassische verteilte Transaktionen (*flache Transaktionen*), die zum Beispiel durch das *Zwei-Phasen-Commit-Protokoll* gesteuert werden, können in Workflow-Systemen nur dann eingesetzt werden, wenn eine Transaktion kurzlebig ist, keine eigenen Subtransaktionen umfasst sowie nur einzelne und unabhängige Datenbanken manipuliert. Solche Transaktionen

werden, da die ACID-Eigenschaften erfüllt sind, auch *Atomare Transaktionen* genannt. Im Gegensatz dazu bezeichnen *Business-Transaktionen* langlebige Transaktionen, die eine hohe Anzahl an Ressourcen binden und dabei mehrere elementare Aktivitäten beziehungsweise atomare Transaktionen umfassen, die jeweils ihre Ergebnisse bereits nach ihrer Beendigung festschreiben dürfen und deshalb meist nur durch Kompensation wieder rückgängig gemacht werden können. Dies bedeutet, dass bei Business-Transaktionen noch während ihrer Laufzeit Effekte nach außen dringen können, die dann andere Aktivitäten außerhalb der Transaktion beeinflussen können. Damit ist bei Business-Transaktionen die Isolationseigenschaft meist eingeschränkt. Zudem können sie mehrere Organisationen und damit Verwaltungs- und Vertrauensdomänen umfassen. Zur Umsetzung werden hierbei neben der Kompensation (*Backward Recovery*) Sphären-Konzepte sowie Mechanismen zur *Forward Recovery* – also dem Versuch, gescheiterte Aktivitäten zunächst erneut auszuführen – eingesetzt. Ein erweitertes Modell, das diese Mechanismen zur Definition eines Transaktionskonzepts einsetzt, ist zum Beispiel das der Kompensationssphären. [YLDW05, CCF+05]

Kompensationssphären Eine *Kompensationssphäre* ist eine beliebige Aggregation von Aktivitäten eines Workflows, die eng miteinander gekoppelt sind und in der jede einzelnen Aktivität erfolgreich beendet werden muss, um die gesamte Sphäre erfolgreich zu beenden. Sollte die Ausführung einer Aktivität innerhalb der Kompensationssphäre scheitern, so müssen die Ereignisse aller vorherigen Aktivitäten ebenfalls rückgängig gemacht werden. Dabei wird das Rückgängigmachen durch entsprechende kompensierende Aktivitäten durchgeführt. Eine Kompensationssphäre bietet deshalb lediglich *semantische Atomarität* für die enthaltenen Aktivitäten, ist aber, da sie von einer strikten Isolation der Sphäre und damit lang andauernden Ressourcensperren absieht, auch für zeitlich ausgedehnte Aktivitäten eines Workflows geeignet. [PVZ+06, Gün03, LR00]

Die Kompensation der Sphäre kann je nach Anzahl der kompensierenden Aktivitäten unterschiedliche Granularitäten aufweisen. Eine *globale Granularität* bedeutet dabei, dass es lediglich eine kompensierende Aktivität für die gesamte Kompensationssphäre gibt, die alle enthaltenen Arbeitsschritte auf einmal rückgängig macht. Bei einer *diskreten Granularität* hingegen wird jede Aktivität der Sphäre durch eine entsprechende Kompensationsaktivität einzeln rückgängig gemacht. Um die bisher geleistete Arbeit zu annullieren, muss hierbei das den Prozess ausführende Ma-

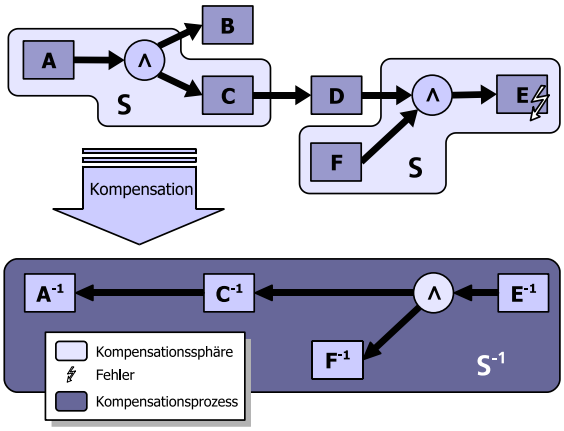


Abbildung 4.14.: Diskretes Zurücksetzen einer Kompensationssphäre (nach (LR00))

nagementsystem, je nach Anzahl und Reihenfolge der bereits erfolgreich beendeten Aktivitäten, einen inversen und damit kompensierenden Teil-Workflow konstruieren und ausführen (Backward Recovery, vgl. Abbildung 4.14). Um jedoch den Verlust bereits erzielter Arbeit zu vermeiden, können Aktivitäten auch als wiederholbar gekennzeichnet werden, sodass vor einer Kompensation zunächst versucht wird, durch erneute Ausführung der zuvor fehlgeschlagenen Aktivität den Fortgang der Sphäre zu sichern (Forward Recovery). [LR00]

Da die Isolationseigenschaft bei Kompensationssphären nicht zwingend gegeben ist, können Sphären interferieren und damit die Weitergabe einer Kompensationsentscheidung notwendig machen. Dabei wird das Weiterreichen des Abbruchs durch Setzen des *Proliferationsattributs* gesteuert. Bei aktivierter Proliferation wird eine abhängige Sphäre ebenfalls zur Kompensation gebracht, da eine oder mehrere ihrer Aktivitäten mit inkorrekten Daten gearbeitet haben. [LR00]

4.3.6. Nicht-funktionale Aspekte

Nicht-funktionale Aspekte ergänzen die Beschreibung des Verhaltens und der Schnittstelle von Systemen – also ihrer funktionalen Eigenschaften –, um Merkmale und Rahmenbedingungen, die während der Ausführung gelten werden oder gelten sollen. Diese Angaben können dabei breit gefächert sein und unter anderem Qualitätsparameter, Angaben über die zeitliche Verfügbarkeit eines Dienstes, Abrechnungsmodalitäten oder Sicherheits- und Vertrauensparameter umfassen. [OEt02, Jab98]

Die Angaben zu nicht-funktionalen Eigenschaften sind dabei besonders bei der Auswahl von Diensten wichtig, da Nutzer meist über die reine

Funktion hinaus Ansprüche an Dienste haben, die erfüllt werden müssen. So kann zum Beispiel ein langsamer, aber dafür kostenloser Dienst einem schnellen, aber teuren Dienst vorgezogen werden, obwohl beide das gleiche (funktionale) Ergebnis für den Nutzer liefern würden. [Sin04]

Prozesse, die nach außen einen Dienst darstellen, können dabei selbst nicht-funktionale Eigenschaften besitzen, die sie dann einem Klienten zusichern. Auf der anderen Seite können sie aber auch selbst als Klient gewisse Eigenschaften von den Diensten verlangen, die zur Ausführung der im Prozess enthaltenen Aktivitäten angesprochen werden. Dazu müssen, gerade bei dynamischer und damit später Bindung aktueller Dienste an die Aktivitäten des Prozesses, die gewünschten nicht-funktionalen Eigenschaften im Workflow modelliert und zur Laufzeit mit den Dienstinstanzen abgeglichen werden.

4.4. Prozessausführung

Die Modellierung eines Geschäftsprozesses als Workflow mit seinen Aktivitäten und Subprozessen sowie dem Daten- und Kontrollfluss (vgl. Abschnitt 4.3) ist eine wichtige Prämisse zu seiner automatisierten Ausführung. Die eigentliche Verwaltung und Verarbeitung konkreter Workflow-Instanzen wird dabei durch ein *Workflow-Management-System* vorgenommen (vgl. Abbildung 4.15). Dieses erzeugt und verwaltet die Ausführung von Workflows mit Hilfe von einer oder mehreren Ausführungskomponenten (*Workflow Engines*), die ein Prozessmodell interpretieren können und die Interaktion mit den Teilnehmern und Diensten übernehmen. [Wor99]

Neben der Kontroll- und Datenflusssteuerung bieten Workflow-Management-Systeme im Allgemeinen zusätzliche administrative und überwachende Komponenten an. Hierzu gehört zum Beispiel das Protokollieren des Workflow-Fortschritts, um in Fehlersituationen eine Wiederaufnahme der Prozessabarbeitung zu ermöglichen oder das Erheben von statistischen Daten, um die Ausführung später analysieren zu können. Die Kernaufgabe bleibt jedoch das Abarbeiten von Workflow-Instanzen, bei dem durch das Workflow-Modell navigiert wird und alle Aktivitäten, die sich auf dem aktuellen Pfad zwischen Start- und Endzustand des Prozesses befinden, ausgeführt werden. [Wor99, Jab97]

4.4.1. Lebenszyklus eines Geschäftsprozesses

Die *Navigation* durch eine Prozessinstanz, zusammen mit dem Abarbeiten der jeweils ausführbaren Aktivitäten, wird in einem Workflow-Manage-

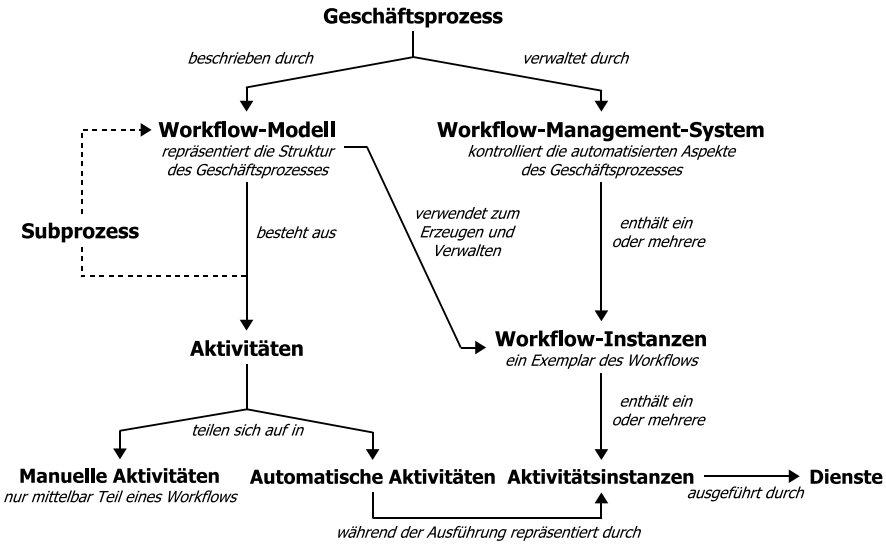


Abbildung 4.15.: Beziehungen zwischen Workflow-Modell und Management (z. T. nach (Wor99))

ment-System von der enthaltenen Workflow-Engine durchgeführt. Hierbei durchläuft der Geschäftsprozess unterschiedliche interne Zustände und Zustandsübergänge, die durch folgenden *Geschäftsprozesslebenszyklus* definiert werden können (vgl. Abbildung 4.16):

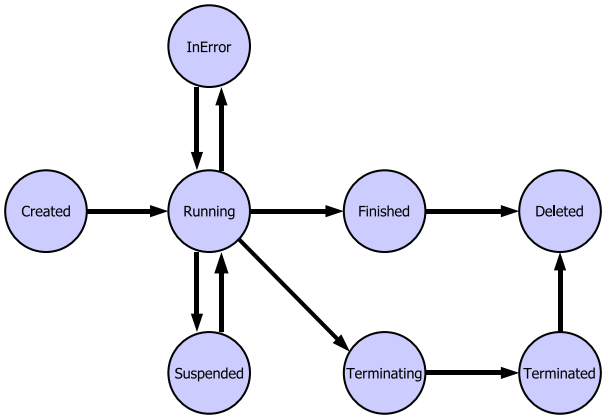


Abbildung 4.16.: Lebenszyklus eines Prozesses (nach (LR00))

Created Unmittelbar nach dem Erzeugen einer Workflow-Instanz auf Basis eines Workflow-Modells befindet sich diese im Zustand *Created*. In diesem Zustand existiert zwar die Prozessinstanz – wird jedoch noch nicht ausgeführt. Im Allgemeinen verweilt der Prozess in diesem Zustand nicht lange, es sei denn, für den Start der Ausführung ist ein in der Zukunft liegender Zeitpunkt als nicht-funktionale Randbedingung angegeben, die die

Ausführung entsprechend verzögert. [LR00, Wor99]

Running Der Hauptzustand in dem sich ein Geschäftsprozess befinden kann, ist der *Running*-Status. In diesem Zustand hat die eigentliche Ausführung der Prozessinstanz begonnen und die Navigation durch den Prozess wird durchgeführt. Dabei wird sukzessive jede Aktivität, die durch den aktuellen Kontroll- und Datenfluss determiniert wird, zur Ausführung gebracht und so der Prozess vorangetrieben. [LR00, Wor99]

Finished In den *Finished*-Zustand geht eine Prozessinstanz dann über, wenn ihre Ausführung ordnungsgemäß abgeschlossen ist und sie damit eine ihrer vorbestimmten Endkonfigurationen eingenommen hat. An dieser Stelle können noch Systemaktivitäten stattfinden, die von der originären Ausführung des Prozesses unabhängig sind – wie zum Beispiel das Schreiben von Protokolleinträgen. [LR00, Wor99]

Suspended Wird eine Prozessinstanz, die gerade abgearbeitet wird, vom Benutzer oder Administrator bewusst unterbrochen, geht sie in den *Suspended*-Status über. In diesem Zustand kommt der Kontrollfluss zum Stillstand und es werden keine weiteren Aktivitäten zur Ausführung gebracht. Der Suspended-Zustand kann nur durch eine explizite Wiederaufnahme der Bearbeitung des Prozesses verlassen werden. [LR00, Wor99]

InError Durch Fehler während der Ausführung kann eine Prozessinstanz in den *InError*-Status übergehen. In diesem Zustand stoppt die Navigation durch den Prozess und es müssen gegebenenfalls von außen Fehlerbehandlungsmaßnahmen eingeleitet werden, um den Prozess anschließend weiter ausführen zu können. [ZK07, LR00]

Terminating Wird eine Prozessinstanz während der Abarbeitung von außen abgebrochen, geht sie in den *Terminating*-Status über. In diesem Zustand werden keine weiteren Aktivitäten mehr gestartet und auf das Ende aller laufenden Aktivitäten gewartet. [LR00, Wor99]

Terminated Der *Terminated*-Zustand gibt an, dass eine Prozessinstanz abnormal beendet wurde. Hierbei wird sie nicht mehr ausgeführt, befindet sich aber auch nicht in einer der vordefinierten Endkonfigurationen. [Wor99]

Deleted Der *Deleted*-Zustand gibt an, dass eine Prozessinstanz aus der Workflow-Engine entfernt wurde. Dabei werden die Ressourcen, die von ihr belegt wurden, wieder freigegeben. [ZK07]

Nicht nur Workflows haben einen Lebenszyklus, auch die einzelnen Aktivitäten, die während der Navigation durch die Prozessinstanz ausgeführt werden, durchlaufen ihren eigenen Lebenszyklus (vgl. Abbildung 4.17). Dieser Zyklus kann dabei durchaus auch Einfluss auf den Lebenszyklus des Workflows im Ganzen haben, da zum Beispiel Fehler bei der Ausführung einer Aktivität zu einem Fehler im Workflow führen können. Die einzelnen Zustände des Aktivitätslebenszyklus haben dabei folgende Bedeutung:

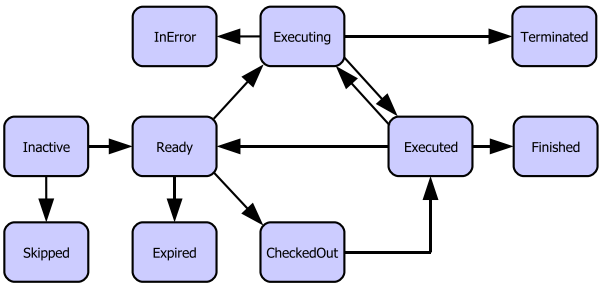


Abbildung 4.17.: Lebenszyklus einer Aktivität (nach (LR00))

Inactive Wird eine Prozessinstanz gestartet, werden alle ihre Aktivitäten in den Zustand *Inactive* versetzt. Damit ist jede Aktivität erzeugt sowie initialisiert und kann, wenn nötig, zur Ausführung gebracht werden. [LR00, Wor99]

Ready Sind alle notwendigen Bedingungen zur Ausführung einer Aktivität erfüllt, wird sie in den Zustand *Ready* versetzt. In diesem Zustand werden alle für die Ausführung notwendigen Vorarbeiten, wie zum Beispiel die Zuordnung von Diensten und Teilnehmern, durchgeführt. [LR00]

Executing Handelt es sich bei der Aktivität um eine automatische Aktivität, geht der Zustand aus dem Ready-Status in den Zustand *Executing* über und der eigentliche Dienstaufruf wird durchgeführt. Dies kann dabei nicht nur einen atomareren Dienst umfassen, sondern auch einen Subprozess, dessen Ausführung durch die Aktivität angestoßen wird. [LR00]

CheckedOut Ist eine Aktivität so modelliert, dass sie nicht automatisch, sondern manuell und damit ohne die Unterstützung des Workflow-Man-

agement-Systems ausgeführt wird, geht sie in den *CheckedOut*-Zustand über. In diesem verweilt die Aktivität, bis die erfolgreiche Ausführung vom durchführenden Teilnehmer bestätigt wurde. [LR00]

Executed Nach erfolgreicher automatischer oder manueller Ausführung geht die Aktivität in den *Executed*-Status über. In diesem bleibt die Aktivität, bis sämtliche Nachbedingungen zu ihrem Verlassen eingetreten sind. [LR00]

Finished Sind alle Endbedingungen der Aktivität erfüllt, geht sie in den Zustand *Finished* über. Die Ausführung der Aktivität ist damit abgeschlossen, und das Ergebnis wird der übergeordneten Workflow-Instanz übergeben, welche die Navigation des Prozesses fortführt. [LR00]

Skipped Können Aktivitäten aufgrund der aktuellen Kontrollflusssituation nicht mehr zur Ausführung kommen, werden sie in den Zustand *Skipped* versetzt. Dies kann zum Beispiel dann vorkommen, wenn ein alternativer Pfad während der Navigation gewählt wurde oder wenn eine Join-Bedingung nicht mehr erfüllbar ist. [LR00]

Expired Wird die Ausführung einer zwar potenziell ausführbaren Aktivität nicht mehr benötigt, wird sie durch Setzen des *Expired*-Status von der weiteren Bearbeitung ausgeschlossen. [LR00]

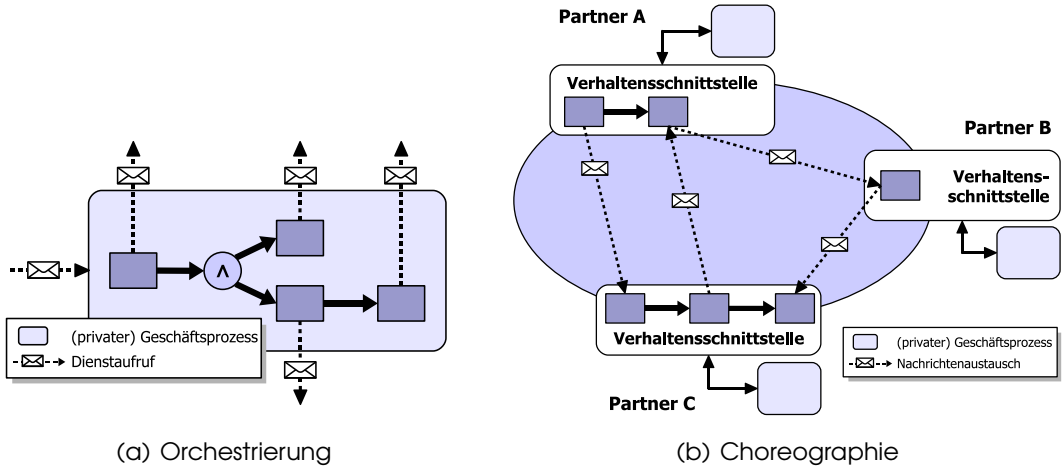
InError Treten bei der Ausführung der Aktivität Fehler auf, geht die Aktivität in den Zustand *InError* über. In solchen Fällen müssen entweder Fehlerbehandlungsmaßnahmen auf Ebene des Workflows vorgenommen werden oder es wird zuvor versucht, durch nochmaliges Ausführen der Aktivität doch noch einen erfolgreichen Abschluss zu erhalten. [LR00]

Terminated Wird die Ausführung einer Aktivität von außen abgebrochen, weil zum Beispiel der übergeordnete Prozess vom Benutzer vorzeitig beendet wurde, geht die Aktivität in den Zustand *Terminated* über und ihre Ausführung bricht ab. [LR00]

4.4.2. Kompositionsparadigmen zur Konstruktion von Prozessen

Dienste für sich betrachtet erbringen zunächst eine isolierte Aufgabe, die keine sichtbaren Beziehungen zu anderen Diensten aufweist (vgl. Ab-

schnitt 4.1). Solche isolierten Dienste sind zwar in vielen Situationen ausreichend, aber ihr Nutzenpotenzial kann sich erheblich steigern, wenn man sie zum Erreichen eines erweiterten Anwendungsziels aggregiert beziehungsweise zu neuen komplexen Diensten komponiert. Dabei beschreibt ein Geschäftsprozess, respektive der entsprechende Workflow, aus welchen Diensten eine komplexe Aufgabe aufgebaut ist und wie diese Dienste koordiniert werden (vgl. Abschnitt 4.2). Je nachdem aus welcher Perspektive ein solcher Geschäftsprozess betrachtet wird und welcher Verteilungsaspekt betont werden soll, unterscheidet man zwischen der *Orchestrierung* und der *Choregographie* von Prozessen. [Ley03, Pel03b]



(a) Orchestrierung (b) Choreographie

Abbildung 4.18.: Kompositionsparadigmen von komplexen Diensten

Orchestrierung Die Orchestrierung komplexer Dienste führt zu ausführbaren Geschäftsprozessen, wobei die zentrale Sicht eines einzigen Beteiligten auf den Prozess repräsentiert wird. Workflows, die nach diesem Koordinationsparadigma modelliert werden, beschreiben sowohl die Kommunikationsbeziehungen als auch die internen Abläufe, welche die am Prozess beteiligten Dienste eingehen (vgl. Abbildung 4.18(a)). Der Prozess ist also klassisch durch seinen Daten- und Kontrollfluss definiert und kann zentral von einer Prozess-Engine ausgeführt werden. Durch diese Zentralisierung ist es leicht möglich, den orchestrierten Prozess selbst wiederum als (komplexen) Dienst anzubieten. [BDO06, Pel03a, Pel03b]

Choreographie Choreographien sind hingegen auf eine Zusammenarbeit der beteiligten Dienste ausgerichtet. Dementsprechend modellieren sie Prozesse auf eine verteilte Art, bei der jeder beteiligte Dienst seinen

Teil zur Choreographie beiträgt. Insgesamt wird der choreographierte Prozess durch die Nachrichtensequenz, die während seiner Ausführung zwischen den Teilnehmern und Diensten auftritt, determiniert (vgl. Abbildung 4.18(b)). Es wird also eine globale Sicht auf den extern sichtbaren Effekt beschrieben, der alle Interaktionen zwischen den Diensten wiedergibt, jedoch keine der internen Aktionen beteiligter Dienste umfasst. Der durch die Choreographie definierte Gesamtprozess ist daher in dieser Form noch nicht ausführbar. [BDO06, Pel03a, Pel03b]

Nimmt ein Dienst oder ein Prozess an einer Choreographie teil, wird durch eine zusätzliche Verhaltensschnittstelle (*Behavioural Interface*) beschrieben, an welchen Interaktionen er innerhalb des Prozesses beteiligt ist. Diese Verhaltensschnittstelle ergänzt somit die funktionale Schnittstelle – welche die Gesamtheit aller möglichen Aufrufe einer Dienstinstanz auflistet – um die Kommunikationsakte, die dieser Dienst als Teilnehmer eines bestimmten Prozesses wahrnimmt. Der gesamte Geschäftsprozess ergibt sich dann (virtuell) aus den Verhaltensschnittstellen aller beteiligten elementaren und komplexen Dienste. [BDO06]

4.5. Sprachen zur Workflow-Modellierung

Die unterschiedlichen Kompositionsparadigmen der Orchestrierung und der Choreographie (vgl. Abschnitt 4.4.2) finden sich auch in den existierenden Sprachen zur Modellierung von Geschäftsprozessen wieder. Stellvertretend für die unterschiedlichen Sprachfamilien werden im Folgenden *WS-BPEL* zur Modellierung von Orchestrierungen von Web Services, *WSCDL* zur Modellierung von Choreographien von Web Services und XPDL als metasprachlicher und technologieneutraler Ansatz beschrieben.

4.5.1. WS-BPEL – Web Service Business Process Execution Language

Die *Web Service Business Process Execution Language* (WS-BPEL) ist eine offene und XML-basierte Sprache zur formalen Definition von Geschäftsprozessen und Interaktionsprotokollen. Sie wird von der *Organization for the Advancement of Structured Information Standards* (OASIS) als Industriestandard herausgegeben und basiert auf der zuvor standardisierten Sprache BPEL4WS (*Business Process Execution Language for Web Services*). WS-BPEL erweitert dabei den Web-Service-Protokollstapel um eine Schicht zur Prozessorchestrierung, sodass mit Web Services auch langlebige Geschäftstransaktionen modelliert werden können. [BBE+07, HCE+06]

WS-BPEL bietet die Möglichkeit, sowohl ausführbare als auch abstrakte und somit nicht ausführbare Geschäftsprozesse zu definieren. Dabei beschreibt ein *ausführbarer Geschäftsprozess* das tatsächliche Verhalten eines Teilnehmers innerhalb einer Geschäftstransaktion. Er stellt also den privaten Workflow dar, den der Teilnehmer ausführt, um den von ihm angebotenen Dienst zu erbringen. Hierbei bestimmen die modellierte Logik und der Zustand des Prozesses den konkreten Ablauf der internen und externen Dienstaufrufe und somit das Interaktionsverhalten innerhalb des modellierten (privaten) Geschäftsprozesses. *Abstrakte Prozesse* hingegen sind als Geschäftsprotokolle definiert, die das Prozessverhalten partiell beschreiben, ohne auf konkrete Details der Ausführung einzugehen. Die nicht ausführbaren Prozesse verknüpfen dabei Schnittstellendefinitionen von Diensten mit Verhaltenbeschreibungen, die von den Teilnehmern in einem übergeordneten Prozess erwartet werden. Um abstrakte Prozesse zur Ausführung zu bringen, ist eine so genannte *ausführbare Erweiterung* notwendig, bei der eine Menge von ausführbaren Geschäftsprozessen beziehungsweise Diensten den abstrakten Teilen der Beschreibung zugeordnet werden. [BBE+07, HCE+06, PD04]

Die Struktur von WS-BPEL-Prozessen

Ausführbare WS-BPEL-Prozesse orchestrieren Web Services zu neuen höherwertigen Diensten, die dann ebenfalls Web Services darstellen. Sie sind in Form von XML-Dokumenten beschrieben und durch eine WSDL-Schnittstelle ansprechbar. Als Wurzel der Prozessdefinition enthält der `process`-Container alle Informationen, die den Prozess beschreiben (vgl. Abbildung 4.19). Er definiert damit zugleich den globalen Sichtbarkeitsraum des Prozesses, der durch die Definition von lokalen Sichtbarkeitsräumen – so genannten `scopes` – weiter strukturiert werden kann. [BBE+07]

Administrative Elemente Über das `import`-Element werden Abhängigkeiten des WS-BPEL-Prozesses mit externen XML-Schemata und WSDL-Beschreibungen deklariert. Daneben werden im `extension`-Element Namensräume für erweiternde Attribute und Elemente, die im Prozess verwendet werden sollen, angegeben. [BBE+07]

Modellierung von Geschäftsbeziehungen WS-BPEL-Prozesse beschreiben die Orchestrierung von Web Services durch die Angabe der enthalte-

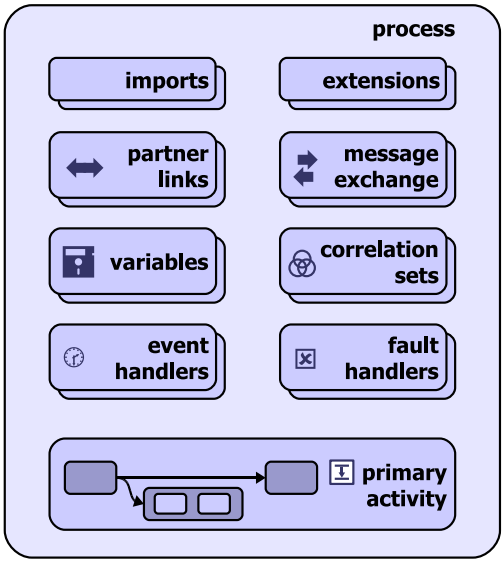


Abbildung 4.19.: Struktur eines Prozesses in WS-BPEL (nach (Kön07))

nen Interaktionen, die notwendig sind, um den Geschäftsprozess erfolgreich auszuführen. Dabei werden diese Interaktionen durch *Partner Links* definiert, welche die Kommunikation der Geschäftspartner untereinander beschreiben. Eine solche Verbindung wird als getypter Konnektor modelliert, der die WSDL Port Types (vgl. Abschnitt 4.1.2), die der Prozess anbietet und von seinen Partnern benötigt, sowie die bei der Kommunikation eingenommenen Rollen der Partner definieren. [BBE+07]

Abbildung des Prozesszustands Der Zustand von WS-BPEL-Prozessen wird durch die jeweiligen Daten des Prozesses bestimmt. Diese werden im Prozess als getypte Variablen angegeben. Ihr Inhalt kann sich dabei entweder durch den Austausch von Nachrichten oder durch prozessinterne und damit private Zwischenergebnisse ergeben. Die Variablen eines WS-BPEL-Prozesses sind deshalb entweder WSDL-Nachrichtentypen oder einfache beziehungsweise komplexe Datentypen der XML-Schema-Definiton. Um die Inhalte der Variablen aus anderen Elementen der Prozessdefinition abfragen oder manipulieren zu können, wird eine *Expression Language* benötigt, die es erlaubt die Elemente eindeutig zu referenzieren. Standardmäßig wird hierzu die Sprache *XPath*[2] verwendet, welche jedoch durch andere Sprachen ersetzt werden kann. [BBE+07]

[2] XPath ist eine Sprache, um einzelne Elemente in einem XML-Dokument anhand eines hierarchischen Pfads adressieren zu können (vgl. `http://www.w3.org/TR/xpath`).

Reaktion auf Ereignisse In WS-BPEL-Prozessen ist es möglich, parallel zum Hauptkontrollfluss auf eintretende Ereignisse zu reagieren. Hierzu können *Event Handler* definiert werden, die durch eingehende Nachrichten oder durch das Ablaufen von Timern aktiviert werden. Dabei können solche Ereignisse keine neuen Prozessinstanzen erzeugen, sondern werden immer von einer bereits laufenden Instanz parallel verarbeitet. [BBE+07]

Reaktion auf Ereignisse Sind Beziehungen zwischen eingehenden Nachrichten und erwarteten Antworten zusammen mit den assoziierten WSDL-Operationen und Partner Links mehrdeutig, müssen durch die Angabe expliziter *Message-Exchange-Attribute* eindeutige Beziehungen hergestellt werden. Diese können dabei wieder für unterschiedliche Sichtbarkeitsräume definiert werden, sodass die unterschiedliche Behandlung einer Nachricht je nach aktuellem Ausführungszustand möglich ist. [BBE+07]

Zuordnen von Nachrichten zu Prozessinstanzen Um eingehende Nachrichten einer Prozessinstanz zuordnen zu können, bietet WS-BPEL die Möglichkeit, *Korrelationsmengen* zu definieren. Diese bauen auf der Annahme auf, dass viele Nachrichten zwischen Dienstanbietern und -nutzern bereits Daten enthalten – zum Beispiel eine Kundennummer oder eine Bestellnummer –, die genutzt werden können, um die Prozessinstanz auszuwählen, für die die Nachricht bestimmt ist. Durch den Abgleich der angegebenen Prozesseigenschaften mit denen, die in der Nachricht enthalten sind, kann dann die Zuordnung der Nachricht erfolgen. [BBE+07]

Behandeln von Fehlersituationen Da während des Ablaufs von Geschäftsprozessen Fehler auftreten können, die es notwendig machen, vom normalen Kontrollfluss abzuweichen, ist es im WS-BPEL-Prozess möglich, durch entsprechende *Fault Handler* auf Ausnahmesituationen zu reagieren. Dabei kann auf unterschiedliche Fehler auch unterschiedlich reagiert werden. [BBE+07]

Durch die Verwendung von lokalen Sichtbarkeitsräumen in Zusammenhang mit lokalen Fault Handlern können auf den Ausführungszustand abgestimmte Fehlerbehandlungen durchgeführt werden. Es ist sogar möglich, durch das Weiterreichen von Fehlern an Kompensationsstrukturen (*Compensation Handler*), Prozesse oder Teile von ihnen durch die Möglichkeit der semantischen Kompensation transaktional abzusichern. [BBE+07]

Beschreiben der Prozesslogik Der Kontrollfluss von WS-BPEL-Prozessen wird durch eine Blockstruktur – also der hierarchischen Verschachtelung von strukturierten und elementaren Aktivitäten – angegeben. Eine strukturierte Aktivität definiert dabei einen Teil der Kontrollflusslogik und im Gegensatz dazu beschreibt eine elementare Aktivität eine einzelne ausführbare Handlungsanweisung. [BBE+07]

Zu den elementaren Aktivitäten zählen dabei das blockierende Warten auf eingehende Nachrichten (`receive`), das Versenden von Antworten (`reply`) sowie das Ausführen einer *Request-Reply-Operation* (`invoke`). Aber auch das Zuweisen (`assign`) und Auswerten (`validate`) von Variablen sowie das Generieren von Fehlermeldungen (`throw, rethrow`), das Kompensieren von Fehlern (`compensate, compensateScope`) als auch der Abbruch des Prozesses (`exit`) gehören zu den elementaren Aktivitäten. [Kön07, BBE+07]

Die Definition der Kontrollflusslogik durch strukturierte Aktivitäten bietet die Möglichkeit, eine Aktivität in eine Sequenz (`sequence`) oder Parallelausführung (`flow`) von Subaktivitäten zu gliedern. Ferner können Aktivitäten zu Schleifen zusammengefasst werden, die ein wiederholtes Ausführen der Aktivitäten ermöglichen (`while, repeatUntil, forEach`). Zudem können in WS-BPEL Entscheidungen innerhalb des Kontrollflusses durch bedingte Entscheidungen (`if-elseif-else`) beschrieben werden. [Kön07, BBE+07]

4.5.2. WS-CDL – Web Services Choreography Description Language

Die *Web Services Choreography Description Language* (WS-CDL) ist, wie die meisten Sprachen im Web-Service-Umfeld, eine XML-basierte Sprache und dient zur Beschreibung der unternehmensübergreifenden Zusammenarbeit von Dienstanbietern auf der Basis eines gleichberechtigten Peer-to-Peer-Ansatzes (vgl. Abschnitt 2.5.1.2). Dabei definiert sie als Sprache zur Umsetzung des Choreographie-Paradigmas Prozesse über den zur Ausführung des Prozesses notwendigen Nachrichtenaustausch. WS-CDL ist ein internationaler Standard und wird vom *World Wide Web Consortium* (W3C) herausgegeben. Dabei ist die Sprache unabhängig von den zur Realisierung der Dienste verwendeten Plattformen sowie Programmiersprachen und -modellen. Sie beschreibt das von außen beobachtbare Verhalten der Teilnehmer, indem die Abfolge und Bedingungen, die während des Nachrichtenaustauschs gelten müssen, von einem globalen Standpunkt aus definiert werden. [Fre06, DPC+05, KBR+05]

Die mittels WS-CDL beschriebenen Prozesse definieren auf einem abstrakten Niveau alle Vereinbarungen, die notwendig sind, damit autonome Dienstleister ein gemeinsames Ziel erreichen können. Damit sind diese Choreographien nicht direkt ausführbar, sondern überlassen die Art und Weise der Ausführung den jeweiligen Teilnehmern. Dabei sind in einer WS-CDL-Beschreibung sowohl die statischen und invarianten Beziehungen zwischen den Teilnehmern und ihren unterschiedlichen Rollen beschrieben als auch das dynamische Verhalten und die zeitlichen Abhängigkeiten, die zum Erlangen des gemeinsamen Ziels notwendig sind. [Fre06, KBR+05]

Die Struktur von WS-CDL-Prozessen

Mit Hilfe von WS-CDL-Beschreibungen können also Prozesse beschrieben werden, die durch die Zusammenarbeit von Prozessbeteiligten ausgeführt werden. Dazu werden die allgemeinen und von außen beobachtbaren Anteile des Prozesses in einer XML-Datei beschrieben, die allen Teilnehmern als eine globale Ausführungsvorschrift dient. Dazu definiert der WS-CDL-Standard ein Dokument mit einem `package`-Wurzelelement, das alle weiteren zur Ausführung des Prozesses notwendigen Subelemente enthält (vgl. Abbildung 4.20). Dabei können die enthaltenen Informationen grob in die Elemente zur Beschreibung globaler Typdefinitionen, der Teilnehmer und ihrer Rollen sowie die Definition der eigentlichen Choreographie unterteilt werden. [KBR+05]

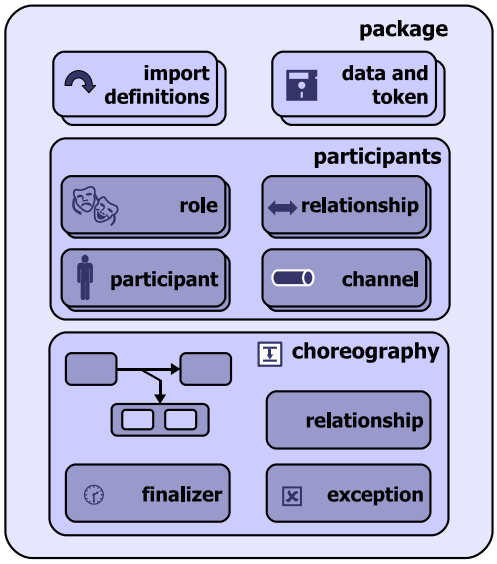


Abbildung 4.20.: Struktur eines Prozesses in WS-CDL

Globale Typdefinitionen Während der Ausführung einer Choreographie wird der Fortschritt zunächst durch den Nachrichtenaustausch zwischen den Teilnehmern beschrieben. Dabei müssen unterschiedliche Teilnehmer ein globales Verständnis der in den Nachrichten ausgetauschten Daten aufweisen. Deshalb ist es möglich, durch die Definition beziehungsweise das Einbinden externer Beschreibungen globaler Datentypen eine für alle Beteiligten zugängliche Typdefinition zu hinterlegen (`informationType`). Um einzelne Teilinformationen eines komplexen Datums adressieren und damit ansprechen zu können, ist es in WS-CDL möglich, so genannte `token` zu definieren. Um dann den Inhalt eines Tokens auch gezielt im Dokument ansprechen zu können, werden `tokenLocator` verwendet, die in Form von XPath-Ausdrücken angegeben werden. [Dub08, KBR+05]

Teilnehmer der Choreographie Zu den statischen Bestandteilen der WS-CDL-Beschreibung zählen Angaben zu den Teilnehmern, die an der Prozessausführung beteiligt sind. Dabei wird zwischen Ausführungsinstanzen (`ParticipantType`) und Rollen (`RoleType`), die von den Teilnehmern in dem beschriebenen Prozess eingenommen werden, unterschieden. Dabei findet jede Interaktion immer zwischen unterschiedlichen Rollen, wie zum Beispiel einem Verkäufer und einem Käufer, statt. Deshalb wird das Verhalten innerhalb des Prozesses als Subelement der Rolle in Form eines `Behavior`-Elements spezifiziert. Zwischen jeweils zwei Rollen und damit zwischen zwei Teilnehmern werden durch das `RelationshipType`-Element die Verantwortlichkeiten innerhalb einer Interaktion bestimmt. [Fre06, KBR+05]

Neben den Rollen und den Beziehungen zwischen den Teilnehmern wird in WS-CDL zusätzlich noch der Kommunikationskanal beschrieben. Hierbei wird die Kanaldefinition durch die Angabe eines `ChannelType`-Elements vorgenommen, das einen Kommunikationsakt zwischen genau zwei Rollen definiert. Damit wird die Verbindung zwischen der Choreographie und der ausführenden Instanz hergestellt. Dies ist im Web-Service-Umfeld im Allgemeinen eine WSDL-Beschreibung (vgl. Abschnitt 4.1.2), kann aber prinzipiell auch andere Beschreibungsformen umfassen. Der so definierte Kommunikationskanal legt fest, welche Informationen in welcher Art und Weise zwischen den Rollen ausgetauscht werden. Zudem kann festgelegt werden, wie oft der Kanal innerhalb einer Kommunikationsbeziehung wiederverwendet werden darf und ob der Kanal auch für mehrere unterschiedliche Aufrufe (*shared channel*) verwendet werden kann. [Fre06, KBR+05]

Beschreibung der Zusammenarbeit Der Kontrollfluss, also die Reihenfolge und die Bedingungen, in der die Interaktionen innerhalb des Prozesses ablaufen, wird durch die Definition von `activity`-Elemente bestimmt, die durch `choreographie`-Elementen zu einer Einheit verbunden sind. Dabei können zwar mehrere Choreographien definiert werden, jedoch kann immer nur genau eine als Wurzelchoreographie und damit Startpunkt des Prozesses ausgezeichnet werden. Damit beschreiben diese Choreographien den dynamischen Teil der Prozessdefinition und können jeweils als Umsetzung eines Anwendungsfalls angesehen werden. [Fre06, KBR+05]

Durch die Angabe von `relationship`-Elementen werden zu der jeweiligen Choreographie die an ihr beteiligten Rollen gebunden. Zusätzlich können Variablen bei einzelnen Teilnehmern definiert werden, welche die Datenabhängigkeiten während der Ausführung des Prozesses beschreiben und von allen Rollentypen eines Teilnehmers gemeinsam zugegriffen werden können (`variableDefinition`). Für die Behandlung von Ausnahme- und Fehlersituationen kann ein `ExceptionBlock` angegeben werden, der Aktivitäten enthält, um auf eine eingetretene Situationen zu reagieren, zum Beispiel in Form einer Kompensation bereits geleisteter Arbeit. Zusätzlich können in einem `FinalizerBlock` Aktivitäten definiert werden, die nach Abschluss der Choreographie ausgeführt werden sollen. [Fre06, KBR+05]

Die einzelnen Arbeitsschritte innerhalb des WS-CDL-Prozesses werden durch Aktivitäten ausgedrückt. Dabei werden *Ordering Structures*, *Work-Unit Notations* und *Basic Activities* unterschieden, um den Kontrollfluss und den Nachrichtenaustausch zu regeln. Hierbei sind die Ordering Structures Behälter für andere Aktivitäten, die jeweils durch ein Ordnungskriterium bestimmt sind. Als Strukturierungsmöglichkeiten bietet WS-CDL die Sequenz (`sequence`), um lineare Abfolgen zu beschreiben, `parallel`-Elemente zur nebenläufigen Ausführung von Aktivitäten und `choice`-Elemente, um Entscheidungen auf der Basis des bisherigen Prozessverlaufs zu treffen. Durch eine WorkUnit werden innerhalb des Kontrollflusses Folgen von Aktivitäten zusammengefasst und durch das Hinzufügen einer Fallunterscheidung (*Guard*) bedingt ausgeführt. Mit diesen bedingten Ausführungsblöcken können zudem Wiederholungen innerhalb der Prozessbeschreibung definiert werden, wobei die entsprechende Abbruchbedingung der Schleife als Bedingung angegeben wird (*Repeat*). Zu den Basic Activities werden alle Aktivitäten gezählt, die atomare Vorgänge repräsentieren. Hierzu zählt die Interaktion (`interaction`), die den koordinierten Nachrichtenaustausch und die Synchronisation von Zustandsin-

formationen zwischen den Teilnehmern regelt. Desweiteren können durch `assign`-Elemente den in WS-CDL definierten Variablen Werte zugewiesen werden und durch `perform`-Elemente Subchoreographien aus einer Elternchoreographie heraus aufgerufen werden. Insgesamt entsteht so ein hierarchischer Gesamtprozess, der kooperativ durch die Prozessbeteiligten ausgeführt wird. [Fre06, KBR+05]

4.5.3. XPDL – XML Process Definition Language

Die *XML Process Definition Language* (XPDL) ist ein Teil des *Workflow Reference Models* der *Workflow Management Coalition* (WfMC) und ist zum Austausch von Prozessdefinitionen entwickelt worden. XPDL definiert als Prozess-Metamodel die elementaren Konstrukte zur Definition von Prozessen, wobei an dieser Stelle auf die Workflow Process Definition fokussiert wird, da dieses Element des Gesamtmodells in einer graphorientierten und XML-basierten Sprache die eigentlichen Prozesse beschreibt. Zwar liegt XPDL bereits in einer zweiten Version vor, jedoch soll hier, da sich im späteren Teil auf die Version 1.0 bezogen wird, diese Version beschrieben werden. [ZK07, MN02]

Die Elemente der XPDL-Prozessdefinition beinhalten zum einen kontextdefinierende Informationen wie Teilnehmer, Daten und Anwendungen sowie zum anderen die über Aktivitäten und Transitionen definierte Ausführungsfolge der Arbeitschritte des Prozesses. Damit enthält das XPDL-Modell alle gängigen Elemente, um Prozesse beschreiben und ausführen zu können. [ZK07, MN02]

Die Struktur von XPDL-Prozessen

Mittels des XPDL-Metamodells ist es möglich, Prozesse abstrakt zu beschreiben und auszutauschen. Grundlage für die Beschreibung des einzelnen Prozesses ist die *Workflow Process Definition* (`WorkflowProcess`), die als Elternelement dient und alle Elemente eines Prozesses enthält (vgl. Abbildung 4.21). Dabei können die notwendigen Angaben zu Prozessteilnehmern, Daten und Anwendungen entweder aus externen Quellen importiert, global in einem mehrere Prozesse zusammenfassenden Paket (`Package`) definiert und dann implizit ver- beziehungsweise ererbt werden oder aber diese Angaben werden individuell in der einzelnen Workflow Process Definition gemacht. Insgesamt entstehen so unterschiedliche Sichtbarkeitsräume (*Scope*), mit deren Hilfe globale Definitionen mit lokalen überschrieben werden können. Zusätzlich können in der Work-

flow Process Definition administrative und ausführungsrelevante Angaben gemacht werden, die zum Beispiel Auskunft über den Erstellungszeitpunkt oder den Autor beziehungsweise über initiale Parameter, Zeitbeschränkungen oder Prioritäten geben. [MN02]

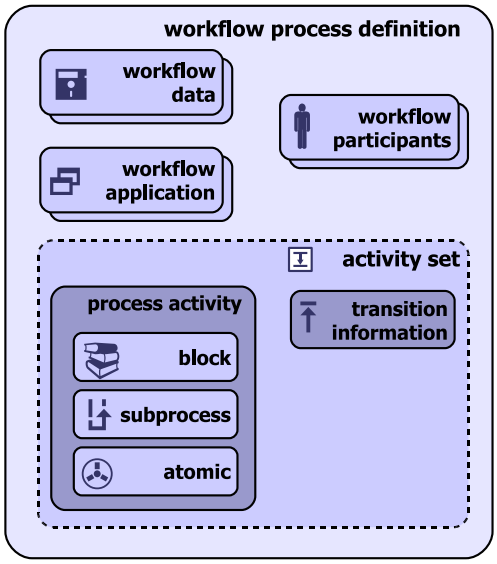


Abbildung 4.21.: Struktur eines Prozesses in XPDL

Teilnehmer des Workflows Die Teilnehmer innerhalb eines XPDL-Prozesses werden durch `Participant`-Elemente definiert, die Ressourcen beschreiben, welche die unterschiedlichen Aktivitäten ausführen können. Diese Teilnehmer müssen dabei nicht unbedingt konkrete Personen oder Geräte sein, sondern können sich auch auf Gruppen beziehen, die gemeinsame Fähigkeiten oder Verantwortlichkeiten aufweisen. Hierzu unterscheidet XPDL Ressourcen und Ressourcenmengen, Organisationseinheiten, Rollen, Personen sowie Systeme. Dabei werden die abstrakten Definitionen erst zur Laufzeit ausgewertet und dann an passende konkrete Personen oder Instanzen gebunden. Damit bietet XPDL ein sehr flexibles Teilnehmermodell, um allgemeine und übertragbare Prozesse zu definieren. [ZK07, MN02]

Beschreibung der relevanten Anwendungen Durch die *Workflow Application Declaration* (`Application`-Element) werden in XPDL-Prozessen alle Anwendungen und Schnittstellen beschrieben, die während der Ausführung aufgerufen werden können. Dabei werden die Angaben auf einem abstrakten Niveau gemacht, um den jeweiligen Prozess auf verschiede-

nen Plattformen mit Hilfe unterschiedlicher Dienstinstanzen ausführen zu können. Auch hier wird erst zur Laufzeit entschieden, welche konkreten Implementierungen zur Ausführung des geforderten Dienstes tatsächlich angesprochen werden. Um dabei Daten des Prozesses an die Schittstelle der Dienste binden zu können, werden bei der Definition der Anwendungen ihre Signaturen mit angegeben (*formale Parameter*), an die dann später konkrete Prozessdaten gebunden werden können. [MN02]

Beschreibung der Prozessdaten Die für die Ausführung eines XPDL-Workflows benötigten Variablen, welche die relevanten Daten des Prozesses repräsentieren, werden in XPDL durch die Angabe einer Liste mit Datenfeldern beschrieben (`DataFields`). Diese prozessrelevanten Daten werden während der Ausführung des Workflows den Aktivitäten und Anwendungen zugänglich gemacht, um Informationen und Zwischenresultate persistent zu machen sowie Entscheidungen innerhalb des Kontrollflusses zustandsabhängig treffen zu können. Dabei müssen alle Datenfelder beschrieben werden, die potenziell während eines Ablaufs des Prozesses verwendet werden können. Zudem muss jedem dieser Datenfelder ein Datentyp zugeordnet werden, der von klassischen vordefinierten Datentypen, wie Zeichenketten, Fließkommazahlen oder Datumswerten, über komplexe und aggregierte Datentypen bis zu extern referenzierten Datentypen reichen kann. [MN02]

Kontrollfluss des Workflows Der Kontrollfluss innerhalb eines XPDL-Prozesses wird durch die Angabe von Aktivitäten (`Activity`) und Transitionen (`Transition`) beschrieben und stellt damit einen graphbasierten Modellierungsansatz dar. Jede der in einem Prozess enthaltenen Aktivitäten wird als eine logische und in sich geschlossene Einheit angesehen, die einen Arbeitsschritt des Prozesses repräsentiert. Dabei unterscheidet XPDL verschiedene Arten von Aktivitäten, die unterschiedliche Funktionalitäten abbilden (vgl. Abbildung 4.22). Hierbei stellt die *generische Aktivität* einen einfachen Dienstaufruf dar, der einen Arbeitsschritt des Prozesses ausführt. So genannte *ROUTE-Aktivitäten* sind hingegen Aktivitäten ohne Bindung an einen konkreten Arbeitsschritt und dienen lediglich zum Aufbau komplexer Kontrollflussstrukturen. Um eine Gruppe von Aktivitäten (*Activity Set*) aus dem Kontrollfluss heraus aufrufen zu können, werden im XPDL-Modell *BLOCK-Aktivitäten* verwendet. Durch die Einführung dieser Indirektion wird die Wiederverwendbarkeit einzelner Prozessabschnitte erhöht, sodass insgesamt kompaktere Pro-

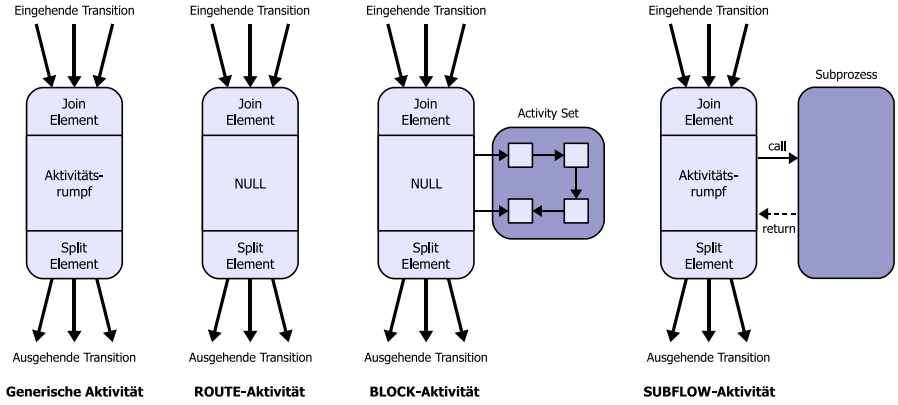


Abbildung 4.22.: Aktivitäten und Transitionsbedingungen (nach (MN02))

zessbeschreibungen möglich sind. Zudem ist es möglich durch *SUBFLOW-Aktivitäten* auf Unterprozesse zuzugreifen, wobei sowohl synchrone wie auch asynchrone Aufrufe abgebildet werden können. [ZK07, MN02]

Um Aktivitäten untereinander zu vernetzen und so den Kontrollfuss abschließend festzulegen, werden in XPDL Transitionen zwischen den einzelnen Aktivitäten definiert. Dabei besitzt jede Aktivität so genannte *Join-Elemente*, die eingehende Transitionen verarbeiten können, und *Split-Elemente*, die ausgehende Transitionen aktivieren können. Dabei unterstützt XPDL sowohl *AND-Splits und -Joins* als auch *XOR-Splits und -Joins*, die durch die oben beschriebenen ROUTE-Aktivitäten zu komplexeren Strukturen verknüpft werden können. Dabei kann jeder Transition zusätzlich eine Bedingung zugeordnet werden, die angibt, wann dieser Übergang beschritten werden darf. Hierbei können sich die Bedingungen der Transitionen sowie die der Join- und Split-Elemente auf die Datenfelder des Prozesses beziehen, sodass das Navigieren durch den Aktivitätsgraphen zustandsabhängig erfolgen kann. [ZK07, MN02]

4.6. Implikationen für diese Arbeit

Das Kapitel hat gezeigt, dass die dienstorientierte Sichtweise auf Softwaresysteme die technologieunabhängige Beschreibung von Aufgaben ermöglicht und inhärent eine lose Kopplung zwischen Softwarekomponenten bietet. Durch die Kapselung von Funktionalitäten in Diensten ist es dann möglich, unterschiedliche Implementierungen derselben Funktionalität zu realisieren und erst zur Laufzeit zu entscheiden, welche Dienstinstanzen die Aufgabe dann tatsächlich ausführen.

Durch die Verknüpfung von Diensten ist es zudem möglich, komplexe

Aufgaben zu realisieren und diese dann selbst wieder als Dienst anzubieten. Hierzu werden die einzelnen Teilaufgaben in einem Prozess zusammengeführt und durch Daten- und Kontrollflusselemente zur neuen komplexeren Aufgabe verbunden. Durch Prozesse beziehungsweise Workflows – also Geschäftsprozesse, die computergestützt ausgeführt werden – können durch die abstrakte Definition der Gesamt- und Teilaufgaben und die späte Bindung von Diensterbringern besonders flexible Kooperationsformen realisiert werden. Hierbei bieten Workflows durch die Möglichkeit zum Einbinden von nicht-funktionalen Rahmenbedingungen eine Steuerung der Ausführung des Prozesses über die reine Geschäftslogik hinaus. Zur Komposition von Workflows gibt es dabei unterschiedliche Ansätze, die auf der einen Seite eine zentrale Sicht auf den Geschäftsprozess ermöglichen (Orchestrierung) oder auf der anderen Seite die unterschiedliche Sicht der Partner einer Kooperation dokumentieren (Choreographie).

Insgesamt kann festgehalten werden, dass mobile Umgebungen aufgrund ihrer inhärenten schwächeren Leistungsfähigkeit und Heterogenität gegenüber klassischen Verteilten Systemen besonders von zusätzlichen Diensten in ihrer Umgebung profitieren können. Dabei bietet gerade die Möglichkeit zur kooperativen Ausführung komplexer Aufgaben als Workflows einen Erfolg versprechenden Ansatz zur besseren Ausnutzung der Potenziale einer mobilen Systemumgebung. Dennoch wird diese Vorgehensweise bisher kaum genutzt und mobile Geräte und Teilnehmer lediglich in bestehende zentral gesteuerte Prozesse integriert. Ein auf die Verhältnisse in mobilen Umgebungen abgestimmtes Konzept zur Ausführung von Workflows als substanzielle Erweiterung der Fähigkeiten mobiler Geräte fehlt jedoch. Deshalb führt das nächste Kapitel das Konzept der *Mobilen Prozesse* ein, mit dessen Hilfe eine kontextbasierte Kooperation innerhalb einer mobilen Umgebung ermöglicht wird, um komplexe Aufgaben benutzerorientiert auszuführen.

5. Migrierende und verteilt ausgeführte Prozesse im Mobile Computing

Es hat sich gezeigt, dass mobile Systeme und damit der Bereich des *Mobile Computing* ein wichtiger Bestandteil aktueller und zukünftiger Innovationen in der Informations- und Kommunikationstechnologie sind (vgl. Kapitel 2). Aufgrund der intrinsischen Eigenschaften mobiler Umgebungen kann jedoch allgemein festgestellt werden, dass diese nicht den gleichen Grad an Verteilungstransparenz bieten können, der in traditionellen Verteilten Systemen möglich ist. Im Gegenteil, die im Vergleich zu stationären Geräten *eingeschränkten Ressourcen*, die erhöhten *Schwankungen der Performanz und Zuverlässigkeit* drahtloser Verbindungen, die *endlichen Energiereserven*, auf die mobile Geräte angewiesen sind sowie die Gefahren, die durch die Mobilität selbst entstehen, führen zu der Erkenntnis, dass mobile Systeme sich ihrer Mobilität bewusst werden und sich ihr anpassen sollten (*Awareness* und *Adaptability*). Zudem besteht im Umfeld mobiler Systeme eine hohe Heterogenität der eingesetzten Hard- und Software, der Betriebssysteme sowie der verfügbaren Kommunikationsprotokolle und -verfahren, sodass oftmals eine Vielzahl unterschiedlicher Geräte und Dienste in mobilen Umgebungen integriert werden müssen.

Um die inhärenten Anforderungen der Mobilität und Heterogenität zu überwinden und einheitliche Systemplattformen für mobile Anwendungen zu schaffen, wird das aus traditionellen Verteilten Systemen bekannte Middleware-Konzept auch auf mobile Umgebungen angewendet (vgl. Kapitel 3). Dabei muss jedoch eine *Middleware für mobile Systeme* neben klassischen Eigenschaften, wie dem Zugriff auf entfernte Ressourcen, der Offenheit und Skalierbarkeit sowie einer (eingeschränkten) Verteilungstransparenz, weitere Anforderungen, wie die Integration von Awareness und Adaptability, eine leichtgewichtige Architektur, die Unterstützung von logischer und physikalischer Mobilität, eine adäquate asynchrone Kommunikation sowie eine möglichst hohe Energieeffizienz aufweisen. Hiermit soll dann die in mobilen Systemen oftmals vorhandene Lücke zwischen den von einer Anwendung benötigten und den durch das eigene Gerät bereitgestellten Ressourcen überwunden werden, um so die erfolgreiche Ausführung von Anwendungen zu ermöglichen.

Durch den Einsatz von Middleware-Plattformen können also mobile Geräte Ressourcen und Dienstleistungen nutzen, die sie selbst nicht bereitstellen können, und so ihre Fähigkeiten erweitern. Daraus ergibt sich, dass mobile Umgebungen, die ihren Anwendungen eine möglichst große Anzahl der in ihnen potenziell vorhandenen Dienste und Ressourcen zugänglich machen, besonders profitabel für das Gesamtsystem sind.

5.1. Middleware-Klassen im Mobile Computing

Je nachdem in welchem Maß eine Middleware im Mobile Computing Anwendungen durch allgemeine Dienste über die Abstraktion der Netzwerkprogrammierung hinaus unterstützt, kann sie einer der Klassen der *kommunikationsorientierten* oder der *anwendungsorientierten Middleware* zugeordnet werden (vgl. Kapitel 3). Dabei bieten kommunikationsorientierte Middleware-Umgebungen schwerpunktmäßig eher systemnahe Basisdienste an, die durch Maskierung des Netzwerkzugriffs den Grad der Verteilungstransparenz im Gesamtsystem erhöhen. Diese Systeme setzen damit in mobilen Umgebungen vor allem klassische Anforderungen an Middleware-Systeme nach *Ressourcenteilung*, *Offenheit*, *Skalierbarkeit*, *Transparenz* und *Fehlertoleranz* auf der Basis asynchroner Kommunikationsformen um. Anwendungsorientierte Middleware-Plattformen nutzen ihre systemnahen Basisdienste hingegen, um darauf aufbauend weiter gehende und abstraktere Dienste zur Unterstützung mobiler Anwendungen zu realisieren. Hierdurch können diese Middleware-Systeme dann auch erweiterten Anforderungen gerecht werden. Hierzu zählt aktuell vor allem die Bereitstellung von Kontextdaten (*Awareness*), damit Anwendungen sich der Umgebung bewusst werden und sich anpassen können (*Adaptability*).

Diese oben genannten Klassen unterscheiden sich dabei vor allem durch die *Granularität* und *Dauer* ihrer angebotenen Systemunterstützung sowie in der *Kopplung* der bereitgestellten Dienste mit den mobilen Anwendungen (vgl. Tabelle 5.1). Hierbei konzentrieren sich kommunikationsorientierte Middleware-Plattformen vor allem auf die zeitliche Entkopplung der Kommunikation zwischen den sie nutzenden (mobilen) Geräten und Anwendungen durch asynchrone Kommunikationsparadigmen (vgl. Abschnitt 3.2). Hierunter fallen unter anderem nachrichtenorientierte, ereignisbasierte und Tuple-Space-basierte Systeme, deren Unterstützung meist auf einzelne und somit isolierte Kommunikationsakte beschränkt ist. Ihr Dienst ist damit im Verhältnis zur Gesamtlaufzeit einer Anwendung auf

eher separate und kurze Zeitintervalle bezogen. Zudem ist die Komplexität der gebotenen Dienstleistung meist gering, was dazu führt, dass die Kopplung zwischen den Anwendungen und dem Middleware-System eher lose ist.

Anwendungsorientierte Middleware-Systeme hingegen stellen erweiterte Dienste bereit, die stärker mit der Anwendung verwoben sind und auch direkten Einfluss auf die Funktionalität haben können. Damit entsteht zwischen der Anwendung und der unterstützenden Middleware-Plattform eine engere Kopplung. Solche Middleware-Systeme bieten dabei meist ein auf eine Applikation oder Anwendungsdomäne abgestimmtes Diensteportfolio. So bietet zum Beispiel die Gaia-Middleware eine Plattform für Anwendungen in Active Spaces, Nexus dient als Basis ortsbasierter Anwendungen und das Context-Toolkit unterstützt die Modellierung von anwendungsspezifischen Kontexten und deren Integration in entsprechende kontextbezogene Applikationen (vgl. Abschnitt 3.3.3). Durch diese Verzahnung von Middleware und Anwendung ist die Dauer der Interaktion auf die gesamte Laufzeit der einzelnen Anwendung angelegt, wobei jedoch die mobilen Anwendungen selbst den Nutzern bei eher kurzzeitigen und einfachen Aufgaben behilflich sind. So werden zum Beispiel Restaurantempfehlungen für die nähere Umgebung gegeben, kontextbezogene elektronische Museumsführer bereitgestellt oder es wird das Abspielen von Videosequenzen automatisch an die sich verändernden Übertragungsbedingungen in mobilen Umgebungen angepasst (vgl. Abschnitt 3.3.1). Dabei sind die Anwendungen eher monolithisch aufgebaut, nutzen ihr Kontextwissen zum Anpassen ihrer Funktionalität sowie zur Auswahl von einfachen Diensten in der Umgebung. Sie interagieren mit entfernten Diensten dann meist nur, um sie als zusätzliche Informationsquelle zur Parametrisierung der eigenen Funktionalität zu nutzen.

Klasse	Granularität	Kopplung	Dauer
anwendungsorientiert	einfache Anwendung	mittel	mittel
kommunikationsorientiert	Kommunikationsakt	gering	kurz
benutzerorientiert	komplexe Anwendung	hoch	lang

Tabelle 5.1.: Eigenschaften der Middleware-Klassen im Mobile Computing

Zusammengefasst gilt, dass die meisten aktuellen Middleware-Plattformen im Mobile Computing eher ad-hoc-statische, monolithisch struktu-

rierte und geschlossene Anwendungen unterstützen (vgl. Abschnitt 3.3.1). Dabei sind sie vorwiegend anwendungszentriert und orientieren sich an momentanen und somit relativ kurzzeitigen Aufgaben und Bedürfnissen mobiler Nutzer. Damit sind sie jedoch vergleichsweise ungeeignet für besonders langlebige und komplexe Aufgaben, vor allem dann, wenn diese zudem über mehrere (mobile) Geräte verteilt sein können. Infolgedessen ergibt sich, dass eine Unterstützung solcher komplexen Anwendungen zurzeit kaum gegeben ist und dadurch ein wichtiger Baustein hin zum *„globalen ubiquitären Computer“*[1] fehlt. Es muss also eine Erweiterung der Middleware-Klassen hin zu einer benutzerorientierten Sichtweise vorgenommen werden, die den Benutzer nicht nur bei einzelnen kurzzeitigen, sondern auch bei allgemeinen langlebigen Aufgaben unterstützt (vgl. Tabelle 5.1). [KZL07b]

5.1.1. Benutzerorientierte Erweiterung

Die fehlende Unterstützung langlebiger und komplexer Aufgaben in aktuellen mobilen Middleware-Plattformen ist vor allem durch die Anwendungsorientierung bestimmt. Diese führt im Allgemeinen dazu, dass die bestehenden Konzepte spezialisiert sind, um gezielt eine Anwendung beziehungsweise Anwendungsdomäne zu unterstützen. Ein Sich-Lösen von der Anwendungsorientierung hin zur Unterstützung mobiler Nutzer durch Middleware-Umgebungen zur Realisierung ihrer komplexen, möglichst sogar erst ad-hoc spezifizierten und langlebigen Aufgaben kann also bisher nicht genutzte Potenziale mobiler Systeme erschließen.

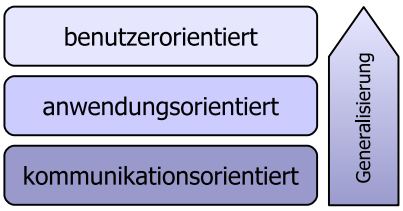


Abbildung 5.1.: Generalisierungsebenen von Middleware-Systemen

Die Einführung einer neuen Klasse *benutzerorientierter Middleware-Systeme* als weitere Generalisierung des Middleware-Ansatzes im Mobile Computing führt dementsprechend dazu, dass der Nutzer mit seinen (spontanen) Aufgaben und Wünschen in den Fokus rückt (vgl. Abbildung 5.1). Mit der Benutzerorientierung soll sich das mobile Gerät von einer

[1] vgl. Kapitel 2

reinen Ausführungsplattform einzelner Anwendungen hin zu einem Zugangspunkt zu den Potentzialen mobiler Umgebungen wandeln, um Anwender bei ihren aktuellen Aufgaben zu unterstützen. Hierbei ist der Nutzer meist an einem Endergebnis beziehungsweise einigen spezifischen Effekten, die mit der komplexen Aufgabe verbunden sind, interessiert und weniger an der konkreten Art wie die Ergebnisse zustande kommen – solange die von ihm gewünschten nicht-funktionalen Rahmenbedingungen eingehalten werden. Solche Rahmenbedingungen können dabei zum Beispiel Kostengesichtspunkte, Qualitätsparameter oder Sicherheits- und Vertrauensaspekte umfassen. [KZL06, Kun05b]

Anforderungen benutzerorientierter Middleware-Systeme Die Anforderungen, die aus dieser neuen Klasse von Middleware-Systemen erwachsen, sind dabei vielschichtig (vgl. Tabelle 5.2). Zunächst müssen komplexe und damit unter Umständen langlebige Aufgaben unterstützt werden (MW1). Diese sind zudem im Allgemeinen a priori unbekannt, sodass ein möglichst generisches Konzept zur Beschreibung der Aufgaben und ihrer Ausführung notwendig ist (MW2). Zudem müssen möglichst die gesamten Potenziale der mobilen Umgebung zugänglich gemacht werden (MW3), wobei insbesondere die Heterogenität und Ressourcenarmut der einzelnen Geräte sowie die unzuverlässigen Kommunikationsverbindungen berücksichtigt werden müssen (MW4).

5.1.2. Heterogenität als Potenzial mobiler Middleware

Mobile Systeme sind inhärent leistungsschwach sowie ressourcenarm und können deshalb besonders von zusätzlichen Diensten in ihrer Umgebung profitieren. Zudem besteht eine große Heterogenität der Ressourcen, die den einzelnen Teilnehmern des Systems direkt oder indirekt zur Verfügung stehen (vgl. Abschnitt 2.4). Dies begründet sich zum einen durch die unterschiedlichen Fähigkeiten, die die Geräte durch entsprechende Hard- und Software aufweisen und zum anderen durch die oftmals verschiedenartigen Netzwerkzugänge und -reichweiten der einzelnen mobilen Geräte, die somit Zugriff auf eine unterschiedliche Anzahl von (entfernten) Diensten ermöglichen.

Dabei gilt im Allgemeinen, dass einer Anwendung umso mehr unterschiedliche Dienste direkt oder indirekt zur Verfügung stehen, je größer die Heterogenität in der mobilen Umgebung ist. Denn mit der Heterogenität

Kürzel	Titel	Details
MW1	Unterstützung komplexer Aufgaben	Benutzerorientierung bedeutet, den Nutzer in den Mittelpunkt zu stellen. Da dieser nicht nur einfache Aufgaben hat, sondern auch Aufgaben, die aus anderen aggregiert beziehungsweise komponiert werden und damit eher langlebig sind, ist eine Unterstützung komplexer Aufgaben notwendig.
MW2	Abstrakte Beschreibung von Aufgaben	Da Benutzer nicht nur bereits vorab bestimmte Aufgaben besitzen, sondern auch solche, die sich ad-hoc ergeben, ist eine Beschreibung notwendig, die ein sehr hohes Abstraktionsniveau bietet, um komplexe Aufgaben spontan definieren zu können.
MW3	Nutzen möglichst aller Potenziale der Umgebung	Spontan definierte komplexe Aufgaben können Aktivitäten umfassen, die nicht vom Gerät des Nutzers bereitgestellt werden. Deshalb ist es notwendig, während der Ausführung komplexer Aufgaben möglichst alle Dienste und Ressourcen der mobilen Umgebung nutzen zu können.
MW4	Berücksichtigen der Eigenschaften mobiler Systeme	Auch bei der Benutzerorientierung müssen die intrinsischen Eigenschaften mobiler Systeme berücksichtigt werden, damit das System in mobilen Umgebungen ausführbar ist.

Tabelle 5.2.: Erweiterte Anforderungen benutzerorientierter Middleware-Systeme

steigt die Anzahl unterschiedlicher Dienste und damit die Wahrscheinlichkeit, dass eine Anwendung, die Zugriff auf einen entfernten Dienst anstrebt, auch einen passenden Dienst in ihrer Umgebung findet (vgl. Abschnitt 5.4.2). Dies gilt vor allem, wenn die benötigten Dienste von vornherein nicht feststehen und deshalb alle Dienste zunächst mit gleicher Wahrscheinlichkeit von einem System benötigt werden. Diese Eigenschaft findet sich besonders im Fall benutzerorientierter Middleware-Systeme mit ihren komplexen und a priori nicht unbedingt bekannten Aufgaben. Stehen hingegen die benötigten Dienste bereits im Vorfeld fest, so bietet natürlich eine hierauf zugeschnittene Dienstauswahl ein besseres Umfeld.

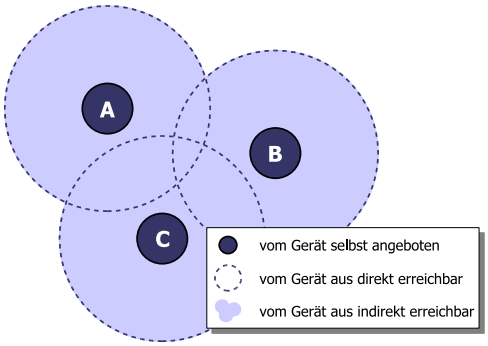


Abbildung 5.2.: Nutzbare Dienste mobiler Geräte

Um nun eine möglichst große Anzahl unterschiedlicher Dienste einer Anwendung in einem mobilen System verfügbar zu machen, reichen kommunikations- und anwendungsorientierte Middleware-Ansätze nicht mehr aus (vgl. Abbildung 5.2). Diese bieten im Allgemeinen eine technologische Isolation des Dienstzugriffs, bei dem eine Anwendung auf eine Middleware-Technologie zum Aufruf entfernter Dienste beschränkt bleibt. Damit bleiben Dienste, die andere in der mobilen Umgebung prinzipiell ansprechbare Geräte aufgrund unterschiedlicher technologischer Voraussetzungen zusätzlich ansprechen können, der lokalen Anwendung verschlossen. Benutzerorientierte Middleware-Plattformen müssen also möglichst eine Brückenfunktion zwischen unterschiedlichen kommunikations- und anwendungsorientierten Middleware-Technologien zum Aufruf entfernter Dienste bieten, damit auch indirekt über andere Teilnehmer erreichbare Dienste zur Erfüllung einer komplexen Aufgabe zur Verfügung stehen und somit die Wahrscheinlichkeit ihrer erfolgreichen Ausführung steigt. Dabei kann durch die Nutzung von Kontextwissen und durch die Kooperation unterschiedlicher Teilnehmer die technologische Isolation aufgehoben werden und so eine neue Klasse von kontextbezogenen Anwendungen entstehen, die sich durch eine *kontextbasierte Kooperation* zur Ausführung *komplexer Aufgaben* auszeichnet.

5.2. Komplexe Aufgaben

Eine einfache beziehungsweise elementare Aufgabe stellt eine klar umrissene und in sich abgegrenzte abstrakte Handlung dar. Sie kann deshalb semantisch nicht in weitere in sich abgeschlossene Teilaktivitäten unterteilt werden. Hierbei ist zu beachten, dass diese Sicht sich nicht auf den strukturellen Aufbau der Implementierung bezieht, der durchaus eine Auf-

teilung in unterschiedliche Komponenten und die Nutzung von externen Diensten vorsehen kann. So sind zum Beispiel ein Druckdienst oder ein Dienst zum Suchen von sich in der Nähe befindenden Restaurants elementare Aufgaben, die eine semantisch abgeschlossene Aktivität beschreiben.

Im Gegensatz hierzu sind *komplexe Aufgaben* eine Komposition aus elementaren Aktivitäten, die gemeinsam eine erweiterte Funktionalität beschreiben und zusammen einen Mehrwert für den Initiator der Aufgabe bieten. Abstrakt können sie als eine Sequenz von in Beziehung stehenden elementaren Aufgaben definiert werden, die ein oder mehrere Ergebnisse beziehungsweise Effekte realisieren. Bei der Ausführung können zudem Zwischenergebnisse auftreten, die jedoch für den Nutzer keine direkte Bedeutung haben und deshalb aus seiner Sicht als untergeordnet betrachtet werden können.

Aufgrund ihres hohen Abstraktionsgrads können durch komplexe Aufgaben individuelle Aktivitäten und Abläufe eines Nutzers abgebildet werden. Damit besitzen sie eine inhärente Benutzerorientierung, die auch nichtfunktionale Anforderungen des Nutzers berücksichtigen kann. Eine solche Ausführung komplexer Aufgaben im speziellen Interesse eines Nutzers verlangt dabei – vor allem bei kooperativen Ausführungsformen (vgl. Abschnitt 5.3) – die explizite Modellierung der nicht-funktionalen Anforderungen und deren Durchsetzung auf allen beteiligten Geräten zur Laufzeit.

Komplexe Aufgaben bieten durch ihre interne Struktur aus abgegrenzten elementaren Aufgaben nahezu ideale Voraussetzungen für ein Umsetzungsmodell, das die Implementierung der einzelnen Aktivitäten von der übergeordneten Anwendungsstruktur und den zur Ausführung notwendigen Daten trennt. Ein solches abstraktes und generisches Modell ist gerade in mobilen Umgebungen, die durch eine hohe Heterogenität in Bezug auf die Leistungsfähigkeit der beteiligten Geräte und deren verfügbaren Ressourcen gekennzeichnet sind, von Vorteil, da durch eine Strategie zum späten Binden von konkreten Implementierungen an die einzelnen Teilaufgaben eine an das jeweilige Gerät und seine Umgebung angepasste Anwendung entstehen kann.

Anforderungen komplexer Aufgaben Systeme, die komplexe Aufgaben in mobilen Umgebungen unterstützen, müssen dazu spezielle Anforderungen erfüllen (vgl. Tabelle 5.3). Zum einen müssen sie zunächst die abstrakte Modellierung der Aufgaben als Komposition von verknüpften elementaren Aufgaben ermöglichen (KA1). Dabei müssen auch nicht-funktionale Anforderungen abbildbar sein, um die Individualisierung auf einzelne Nut-

zer oder Nutzergruppen zu ermöglichen (KA2). Zum anderen muss das Umsetzungsmodell durch eine Trennung der abstrakten Teilaufgaben und einem späten Binden der entsprechenden konkreten Implementierungen den Eigenschaften mobiler Systeme gerecht werden (KA3, KA4).

Kürzel	Titel	Details
KA1	Abstrakte Beschreibung	Die Komposition abgegrenzter einfacher Aufgaben zu komplexen Aufgaben bedingt einen abstrakten Modellierungsansatz zur Beschreibung der Verknüpfung der einzelnen Teilaufgaben zu einem Ganzen.
KA2	Einbinden nicht-funktionaler Anforderungen	Die Benutzerorientierung komplexer Aufgaben erfordert es, dass nicht-funktionale Anforderungen der Nutzer abbildbar sind.
KA3	Trennung von Aufgabe und Implementierung	Um komplexe Aufgaben in mobilen Umgebungen möglichst optimal zu unterstützen, muss die Implementierung von Teilaufgaben entsprechend den Fähigkeiten des ausführenden Gerätes gewählt werden können. Dies bedingt eine Trennung der Aufgabenschnittstelle von ihrer Implementierung.
KA4	Spätes Binden	Durch spätes Binden wird erst ad-hoc zur Laufzeit eine Implementierung an eine Teilaufgabe der komplexen Anwendung gebunden und so eine größtmögliche Flexibilität in der Ausführung erreicht.

Tabelle 5.3.: Anforderungen komplexer Aufgaben in mobilen Umgebungen

5.3. Kontextbasierte Kooperation

Die grundlegende Idee der *kontextbasierten Kooperation* ist, dass in kontextbezogenen Systemen eine Zusammenarbeit auf einer abstrakten technologieunabhängigen Schicht die technologische Isolation mobiler Geräte aufhebt und so zusätzliche Potenziale des gesamten mobilen Systems nutzbar machen. Hierdurch entsteht eine neue Klasse kontextbezogener Systeme, die sich nicht in die etablierten anwendungsorientierten Klassen, wie

die der kontextbezogenen Selektion, Präsentation, Aktion und Annotation (vgl. Abschnitt 3.3.1), einordnen lässt.

Da der Begriff der Kooperation keine etablierte Interpretation aufweist, sondern in unterschiedlichen Disziplinen und Kontexten verschieden gebraucht wird, soll an dieser Stelle zunächst eine eigene Definition für den Kooperationsbegriff eingeführt werden, die dieser Arbeit zugrunde liegt:

> ***Kooperation*** *bezeichnet die selbst koordinierte, gemeinschaftliche Erfüllung von Aufgaben durch zwei oder mehr autonome Subjekte, wobei die kollektive Ausführung einen höheren Grad der Zielerfüllung erreicht als die individuelle Ausführung der Aufgaben.*

Bezogen auf die Ausführung komplexer Aufgaben stellt die Kooperation also eine Zusammenarbeit von unabhängigen Systemen dar, die jeweils Teilaufgaben eigenständig bearbeiten und so das Gesamtziel gemeinschaftlich erreichen. Da dabei die beteiligten Systeme autonom bleiben, gibt es keine zentrale Kontrollstuktur, welche die Ausführung im Ganzen verwaltet (*Peer-to-Peer-Modell*, vgl. Abschnitt 2.5.1.2). Dabei können durch die Delegation von Aufgabenteilen auch solche komplexen Anwendungen erfolgreich ausgeführt werden, die zuvor an den begrenzten Ressourcen und Fähigkeiten eines einzelnen Systems gescheitert wären. Es wird also durch die Kooperation möglich, Nutzen auch aus den Fertigkeiten anderer zu ziehen, was besonders vor dem Hintergrund steigender Anforderungen an mobile Systeme an Bedeutung gewinnt.

Da bei kontextbasierter Kooperation die Ausführung und Verwaltung einer komplexen Aufgabe über mehrere gleichberechtigte Systeme verteilt sind, gewinnen nicht-funktionale Aspekte bei der Ausführung an Bedeutung. Nur durch die explizite Modellierung und das Binden dieser Informationen an die Teilaufgaben ist es möglich, die nicht-funktionalen Bedürfnisse des Initiators auf den autonomen Ausführungssystemen auch durchzusetzen, da eine Interaktion zwischen Ausführungseinheit und Nutzer zur Abfrage von Entscheidungen oder Parametern meist unmöglich ist.

Die Motivation, warum Systeme eine Kooperation eingehen, kann dabei unterschiedlich sein. Zum einen können die Systeme beziehungsweise deren Betreiber gemeinsame Ziele verfolgen und deshalb zusammenarbeiten. Diese gemeinsamen Ziele können dabei monetärer Art sein (*Geschäftsbeziehung*) oder aber rein intentional begründet sein (*Zweckbeziehung*). Auf der anderen Seite kann aber auch die Erwartung stehen, dass die eigene Unterstützung einer fremden komplexen Aufgabe sich dadurch auszahlt,

dass eigene Aufgaben später ebenfalls Unterstützung durch andere finden (*Solidaritätsprinzip*). Diese Beweggründe schließen sich dabei nicht gegenseitig aus, sodass durchaus auch eine Kombination die Motivation zur Kooperation bilden kann.

Anforderungen kooperierender Systeme Kooperierende Systeme im Allgemeinen und kontextbasiert kooperierende Systeme im Speziellen stellen an die sie unterstützenden Middleware-Plattformen und Umsetzungskonzepte erweiterte Anforderungen (vgl. Tabelle 5.4). Dabei bedingt eine kooperative Applikationsausführung ein Aufbrechen monolithischer Anwendungsstrukturen, um Teilaufgaben gemeinschaftlich ausführen zu können (KS1). Dazu muss zunächst die Interaktion mit dem Anwender von der Geschäftslogik so entkoppelt werden, dass die funktionalen Komponenten nur lose an die Darstellungselemente gebunden sind. Dadurch werden direkte Interdependenzen zwischen der Ausführung der Geschäftslogik und der Benutzungsschnittstelle vermieden, die einer Verteilung von Teilaufgaben im Weg stehen würden (KS2). Zudem müssen Geschäftslogik und Interaktions- beziehungsweise Darstellungskomponenten so modelliert und beschrieben werden, dass abgegrenzte und abstrakte Teilaufgaben entstehen, die auf die kooperierenden Systeme verteilt werden können. Die konkrete Umsetzung der Teilaufgaben in ausführbaren Programmcode ist dann den jeweiligen Ausführungssystemen selbst überlassen (KS3). Infolge dessen diffundiert die abstrakte Anwendungslogik mit ihrem Daten- und Kontrollfluss in die ausführende Middleware-Plattform und geht somit eine besonders enge Kopplung mit dieser ein.

Es bedarf also eines Konzeptes, das die verschiedenen Anforderungen der kooperativen Ausführung komplexer Aufgaben in einer mobilen Umgebung vereint. Solch ein Konzept muss dabei gezielt auf die Eigenschaften mobiler Systeme ausgerichtet sein und kann diese dann gegebenenfalls sogar vorteilhaft für das Gesamtsystem nutzen. Das Leitbild, das die geforderten Anforderungen und Eigenschaften aufgreift und dieser Arbeit zugrunde liegt, ist das Konzept der *Mobilen Prozesse*.

5.4. Mobile Prozesse für hoch-dynamische Systeme

Mobile Systeme können – aufgrund des häufigen Ungleichgewichts zwischen benötigten und direkt oder indirekt zugreifbaren Ressourcen – von einer größeren Anzahl von Diensten in ihrer Umgebung profitieren. Dies

Kürzel	Titel	Details
KS1	Entkopplung von Funktionalität und Darstellung	Um eine Kooperation zwischen zwei oder mehreren Systemen zu ermöglichen, müssen Geschäftslogik und Interaktionskomponenten möglichst lose gekoppelt sein. Dies begründet sich vor allem dadurch, dass alle für einen Teilnehmer bestimmten Interaktionen nicht auf fremden und damit dem Teilnehmer nicht zugänglichen Geräten ausgeführt werden können.
KS2	Strukturierung in Teilaufgaben	Eine Aufgabe kann nur dann kooperativ ausgeführt werden, wenn sie aus einzelnen Komponenten aufgebaut ist, die jeweils strukturell unabhängig auf separaten Geräten ausgeführt werden können.
KS3	Trennung von Struktur und Implementierung	Da a priori nicht vorhergesehen werden kann, welche Geräte in einer mobilen Umgebung für die Kooperation vorhanden sind und welche Fähigkeiten sie besitzen, muss die Struktur der Aufgaben möglichst abstrakt und generisch beschrieben werden. Die konkrete Implementierung muss dann zur Laufzeit durch das ausführende Gerät – durch einen lokalen oder entfernten Dienstaufruf – bereitgestellt werden.

Tabelle 5.4.: Erweiterte Anforderungen kooperierender Systeme

gilt insbesondere dann, wenn die mobile Umgebung den Nutzer mit seinen umfangreichen und oftmals langlebigen Aufgaben in den Mittelpunkt stellt (*Benutzerorientierung*). Um diese Aufgaben zu beschreiben, müssen diese *komplexen Aufgaben* modelliert und ausgeführt werden können. Dabei bieten kooperative Ausführungsformen, bei denen die komplexen Aufgaben von unterschiedlichen Systemteilnehmern gemeinschaftlich realisiert werden (*kontextbasierte Kooperation*), ein größtmögliches Potenzial, die Aufgabe erfolgreich abzuschließen. Eine Möglichkeit, die Anforderungen, die sich aus der benutzerorientierten und kooperativen Ausführung komplexer Prozesse ergeben, konzeptuell umzusetzen, sind *Mobile Prozesse*.

5.4.1. Der Begriff des *Mobilen Prozesses*

Mobile Prozesse zeichnen sich dadurch aus, dass sie die Einschränkungen, die sich aus der physikalischen Mobilität von Geräten und Nutzern ergeben (vgl. Abschnitt 2.2.1 und 2.3.2), durch die konzeptionelle Integration von logischer Mobilität relativieren. Hierbei werden durch das Delegieren von Verantwortung für Teile der Ausführung komplexer Aufgaben die Potenziale mehrerer Geräte einer mobilen Umgebung möglichst optimal genutzt und es wird so die Wahrscheinlichkeit der erfolgreichen Ausführung der Aufgaben insgesamt erhöht.

Durch die Analogie in der Definition von komplexen Aufgaben und Prozessen, die beide aus einer Menge von Teilaufgaben bestehen, die in einer logischen Reihenfolge ausgeführt werden (vgl. Abschnitt 4.2 und 5.2), bietet es sich an, komplexe Aufgaben durch (Geschäfts-)Prozesse abzubilden. Zudem bietet eine solche prozessorientierte Modellierung die geforderte Möglichkeit zur abstrakten Beschreibung der Struktur von komplexen Aufgaben sowie deren Trennung von den Details einzelner konkreter Implementierungen.

Eine zentrale Ausführung eines solchen Geschäftsprozesses in einem mobilen Umfeld kann jedoch schnell dazu führen, dass das ausführende mobile Gerät zum Flaschenhals wird und durch seine eigenen Fähigkeiten und Ressourcen die Menge der auf ihm lauffähigen Prozesse begrenzt. Dies ergibt sich zum einen daraus, dass das ausführende Gerät alle zur Ausführung notwendigen Teilaufgaben kennen und lokal oder durch einen entfernten Dienstaufruf ausführen können muss. Zum anderen muss es auch alle Zwischenergebnisse unabhängig von ihrer Größe und Relevanz für die gewünschten Ergebnisse verarbeiten können. Hieraus ergibt sich, dass ein Prozessmodell für mobile Umgebungen die explizite Verteilung der Ausführung und damit kooperative Ausführungsformen berücksichtigen muss, um einen solchen Flaschenhals zu vermeiden. Entsprechend sind *Mobile Prozesse* in dieser Arbeit folgendermaßen definiert:

> *Ein **Mobiler Prozess** ist eine Abfolge zusammengehöriger (z. T. entfernter) Dienste, deren vollständige Bearbeitung i. Allg. eine längere Zeitspanne andauert und sich dabei durch Migration auch über mehrere Geräte erstrecken kann. Die Ergebnisse eines derartigen Prozesses bestehen aus der Summe aller Effekte, die der Initiator im Zuge der Bearbeitung des Gesamtprozesses erwartet.*

Mobile Prozesse dienen also zur Beschreibung komplexer Aufgaben in

mobilen Umgebungen und zu deren kooperativen Abarbeitung. Dabei ist neben der zentralen auch die vollständig verteilte Koordinierung der Ausführung nicht geeignet. Denn durch die hohe Dynamik in mobilen Umgebungen kann es während der Ausführung von (lang andauernden) Aktivitäten zu großen Veränderungen in der Umgebung kommen, sodass die für eine vollständig verteilte Koordination notwendigen stabilen Kommunikationsbeziehungen meist nicht etabliert oder aufrecht gehalten werden können. Deshalb realisiert das Konzept der Mobilen Prozesse eine hybride Koordinationsform, die eine lokale und temporäre zentrale Ausführungskontrolle mit einer verteilten Koordinierung durch ein opportunistisches Verfahren kombiniert. Dabei führt ein Gerät solange die Aktivitäten einer komplexen Aufgabe aus wie es dafür lokale oder entfernte Dienste nutzen kann. Kommt es in diesem Prozess zu Situationen, in denen das aktuell ausführende Gerät, zum Beispiel durch Fehlersituationen oder dem Fehlen von passenden Diensten, die komplexe Aufgabe nicht voranbringen kann, so wird ein anderes Gerät in der mobilen Umgebung gesucht, das durch seine Fähigkeiten die Aufgabe weiter ausführen kann. Dieses Migrieren des Prozesses innerhalb der mobilen Umgebung wird solange weiter betrieben bis die Aufgabe vollständig ausgeführt oder aber endgültig gescheitert ist. Hierbei können alle Ausführungskontexte der verschiedenen Geräte in der mobilen Umgebung für die Ausführung der (komplexen) Aufgabe genutzt werden, sodass durch diese Form der Kooperation ein breiteres Dienst- und Ressourcenangebot gegenüber einem technologisch isolierten Einzelgerät genutzt werden kann (vgl. Abschnitt 5.1.2).

5.4.2. Steigerung der Ausführungswahrscheinlichkeit durch Migration

Dass die Migrationsstrategie zur Ausführung Mobiler Prozesse tatsächlich eine Steigerung der Ausführungswahrscheinlichkeit einer Aufgabe zur Folge hat, kann mathematisch begründet werden. Dabei werden – ohne Beschränkung der Allgemeinheit – folgende Annahmen gemacht:

- Die Wahrscheinlichkeit, mit der ein Gerät eine Aufgabe ausführen kann, ist für jedes Gerät gleich und wird mit p bezeichnet.
- Die Wahrscheinlichkeit, mit der eine Migration zu einem anderen Gerät stattfindet, ist für alle Geräte der mobilen Umgebung gleich und wird mit q bezeichnet.
- Die betrachteten Ereignisse sind stochastisch unabhängig.
- Die Anzahl der Migrationsschritte wird mit n bezeichnet.

- Da nur Aussagen über die Ausführungswahrscheinlichkeit gemacht werden sollen, bleiben Kostengesichtspunkte, die zum Beispiel bei der Migration anfallen, an dieser Stelle unberücksichtigt.

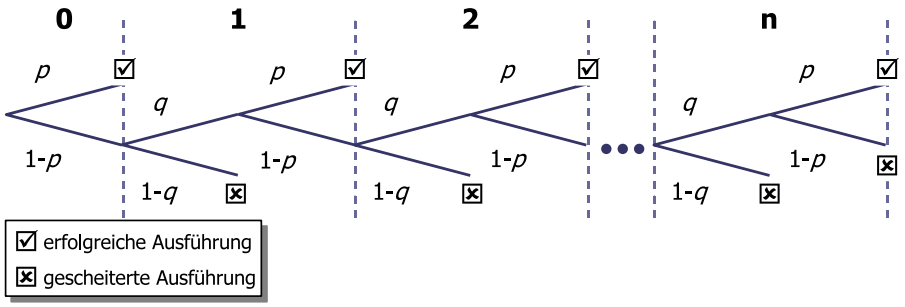


Abbildung 5.3.: Wahrscheinlichkeitsbaum

Mit diesen Annahmen lässt sich ein Wahrscheinlichkeitsbaum aufstellen, der alle Pfade enthält, die zu einer erfolgreichen Ausführung einer Aufgabe führen (vgl. Abbildung 5.3). Hierbei wird zunächst durch das lokale Gerät mit der Wahrscheinlichkeit p die Aufgabe ausgeführt. In den Fällen, in denen die Ausführung scheitert (Wahrscheinlichkeit: 1-p), wird mit der Migrationswahrscheinlichkeit q zu einem anderen Gerät migriert und dort wiederum mit der Wahrscheinlichkeit p die Aufgabe ausgeführt. Diese Schritte werden solange wiederholt, bis die Aufgabe ausgeführt ist oder endgültig scheitert. Hieraus lässt sich dann die Gesamtwahrscheinlichkeit als Summe über sämtliche Erfolgspfade bilden (5.1). Die hieraus abgeleitete Summenformel (5.2) bildet eine *Geometrische Reihe* und lässt sich somit in eine entsprechende *geschlossene Form* (5.3) überführen. Da sowohl die Ausführungswahrscheinlichkeit p als auch die Migrationswahrscheinlichkeit q zwischen Null und Eins liegen, liegt auch die Basis innerhalb der Geometrischen Reihe zwischen Null und Eins, sodass die Reihe konvergiert.

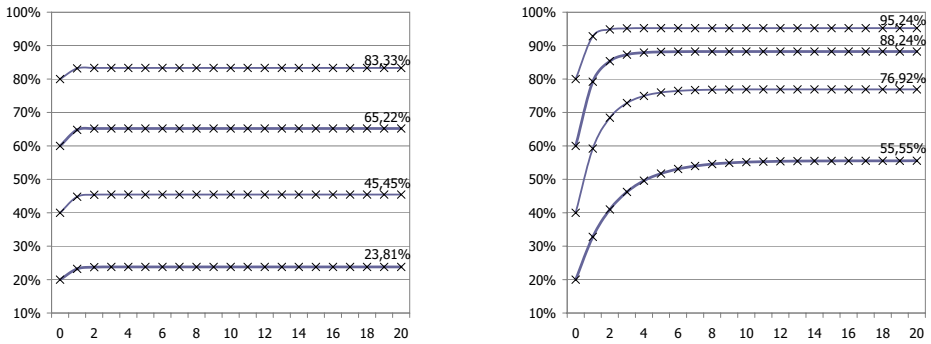
$$P_{Erfolg} = p + (1-p)qp + ((1-p)q)^2 p + \cdots + ((1-p)q)^n p \quad (5.1)$$

$$= p \sum_{i=0}^{n} ((1-p)q)^i \quad (5.2)$$

$$= p \frac{1 - ((1-p)q)^{n+1}}{1 - ((1-p)q)} \quad (5.3)$$

Mit Hilfe der Formel (5.3) lässt sich nun für verschiedene Ausführungs- und Migrationswahrscheinlichkeiten die Gesamtwahrscheinlichkeit einer

erfolgreichen Ausführung berechnen (vgl. Abbildung 5.4). Dabei wird die Aussage, dass die Migration zu einer erhöhten Ausführungswahrscheinlichkeit führt, bestätigt. Zudem wird auch die Aussage über den positiven Effekt der Heterogenität bei der Migration von Mobilen Prozessen belegt.



(a) Migration mit 20%

(b) Migration mit 80%

Abbildung 5.4.: Steigerung der Ausführungswahrscheinlichkeit (vgl. Tabelle A.1)

Ein Ergebnis dieser mathematischen Analyse ist, dass die Ausführungswahrscheinlichkeit tatsächlich durch die Migration steigt. Dabei zeigt sich, dass schon wenige Migrationsschritte zu einer – vor allem im Bezug auf die zu erwartende Grenzwahrscheinlichkeit – deutlichen Steigerung der Ausführungswahrscheinlichkeit führen. So wird zum Beispiel bei einer Migrationswahrscheinlichkeit von 80% und einer Ausführungswahrscheinlichkeit der Geräte von 20% bereits nach zwei Migrationsschritten die Gesamtausführungswahrscheinlichkeit einer Aufgabe mehr als verdoppelt und erreicht bereits etwa 80% der zu erwartenden Grenzwahrscheinlichkeit (vgl. Abbildung 5.4(b)).

Der Einfluss der Heterogenität zeigt sich im Vergleich der erreichbaren Grenzwahrscheinlichkeiten der erfolgreichen Ausführung einer Aufgabe bei unterschiedlichen Migrationswahrscheinlichkeiten (vgl. Abbildung 5.4(a) und 5.4(b)). So liegt zum Beispiel die Grenzwahrscheinlichkeit der erfolgreichen Ausführung bei einer Ausführungswahrscheinlichkeit des einzelnen Geräts von 20% und einer Migrationswahrscheinlichkeit von ebenfalls 20% bei nur ungefähr 24%. Bei einer Migrationswahrscheinlichkeit von 80% hingegen bei etwa 55,5%. Hierbei zeigt sich also, dass je höher der Heterogenitätsgrad und damit die Migrationswahrscheinlichkeit ist, umso größer ist auch die Grenzwahrscheinlichkeit. Bei einer (zumindest theoretisch erreichbaren) Migrationswahrscheinlichkeit von 100% – bei der also für jeden benötigten Dienst auch ein Gerät gefunden werden kann, das diesen bereitstellt – kann sogar eine erfolgreiche Ausführung

garantiert werden. Dies ist dann sogar unabhängig von der Ausführungswahrscheinlichkeit des einzelnen Gerätes möglich. So lässt sich also die Annahme des positiven Einflusses der Heterogenität bei der migrationsbasierten kooperativen Ausführung statistisch belegen.

5.4.3. Eigenschaften Mobiler Prozesse

Mobile Prozesse haben in ihrem Kern einen ähnlichen konzeptionellen Aufbau wie etablierte Workflows. Sie müssen jedoch hierauf aufbauend weitere Eigenschaften bieten, um eine migrationsbasierte kooperative Ausführung in mobilen Umgebungen zu ermöglichen. Dabei sind Mobile Prozesse Workflows, die speziell auf mobile Systeme abgestimmt sind und nicht Geschäftsprozesse, die mobile Geräte lediglich als zusätzlichen Diensterbringer mit einbeziehen. Beide beschreiben hierbei komplexe Aufgaben beziehungsweise Business-Funktionen in Form einer bedingten Abfolge von Teilaufgaben. Sie werden zunächst abstrakt definiert und bei Bedarf werden einzelne Instanzen dieser Prozess-Modelle erzeugt, die dann rechnergestützt ausgeführt werden. In der kooperativen Art, wie Mobile Prozesse ausgeführt werden, liegt einer der elementaren Unterschiede zu traditionellen Workflows, die zentral verarbeitet werden. Mobile Prozesse nutzen dabei die Heterogenität und Dynamik einer mobilen Umgebung, indem durch eine opportunistische Strategie die Ausführung des Prozesses auf unterschiedliche Geräte verteilt wird. Da diese Kooperation unterschiedlicher Teilnehmer einer mobilen Umgebung durch die Migration des Prozesses zwischen den Geräten einer mobilen Umgebung stattfindet, muss – im Gegensatz zum Workflow – auch der Zustand des Prozesses im Prozessmodell beschreibbar sein. Dies ist notwendig, um nach einem Migrationsschritt die zuvor geleistete Arbeit durch die neue Ausführungsumgebung fortführen zu können. Im Einzelnen sehen die Eigenschaften Mobiler Prozesse dabei folgendermaßen aus:

Zunächst muss auch ein Mobiler Prozess die einzelnen Anwendungsschritte (*Aktivitäten*) der durch ihn repräsentierten Aufgabe und deren Ausführungsreihenfolge (*Kontrollfluss*) beschreiben. Dabei müssen die Aktivitäten jedoch möglichst abstrakt beschrieben werden, da a priori – also zur Entwurfszeit des Prozesses – nicht bestimmt werden kann, welche Technologien auf dem später ausführenden Gerät zum Realisieren der Teilaufgabe zur Verfügung stehen werden. Die abstrakte Beschreibung wird auch deshalb benötigt, da zur Laufzeit des Prozesses durch die Migration der Mobilen Prozesse und die hohe Dynamik in der mobi-

len Umgebung erst unmittelbar vor der Ausführung einer Aktivität bestimmt werden kann, welche Dienstinstanzen zur Ausführung erreichbar sind (*spätes Binden*). Zusammengefasst gilt also, dass während der Definition eines Mobilen Prozesses im Regelfall keinerlei Annahmen über das „Wer“, „Wann“ und „Wo“ der Ausführung gemacht werden sollten und teilweise auch nicht gemacht werden können. Deshalb wird eine sprach- und technologieunabhängige abstrakte Beschreibung von Aktivitäten mit einer Strategie zur späten Bindung benötigt (vgl. Abschnitt 5.5.4). Dabei werden die Aktivitäten mittels einer *opportunistischen Strategie* ausgeführt, bei der zunächst lokal nach Dienstinstanzen gesucht wird, welche die auszuführende Aktivität implementieren. Scheitert dies, werden vom Gerät aus erreichbare entfernte Dienstinstanzen gesucht und, sollte auch dies ohne Erfolg bleiben, wird der Prozess auf ein anderes Gerät migriert, um einen neuen Ausführungskontext und damit neue Dienste zur Ausführung nutzen zu können.

Daneben muss auch der Datenfluss zwischen den Aktivitäten definiert werden können und es muss die Möglichkeit bestehen – wenn notwendig – Teilnehmer sowie Rollen anzugeben und den Aktivitäten zuzuordnen. Hierbei ist jedoch zu beachten, dass jegliche Einschränkungen durch die Angabe von konkreten Teilnehmern sich negativ auf die Ausführungswahrscheinlichkeit auswirken können. Zusätzlich muss durch die migrationsbasierte Kooperation auch der konkrete Zustand des Prozesses beschreibbar sein, damit der Mobile Prozess nach der Migration an der Stelle fortgeführt werden kann, an der er zuvor unterbrochen wurde. Insgesamt gilt also, dass die fehlende zentrale Steuerung die Erweiterung der Beschreibung des Prozesses notwendig macht, damit der Prozess zustandsbewahrend in der mobilen Umgebung verteilt ausgeführt werden kann.

Da Mobile Prozesse für mobile Systeme optimal angepasst sein sollen, müssen sie die Anforderungen, die sich aus der Mobilität ergeben, besonders berücksichtigen. Dazu zählt vor allem eine möglichst kompakte Beschreibung, die einfach und ressourcenschonend kommuniziert, interpretiert und ausgeführt werden kann. Zudem muss der höheren Fehleranfälligkeit von mobilen Systemen durch eine erweiterte Fehlerbehandlung Rechnung getragen werden.

Die Migrationseigenschaft von Mobilen Prozessen sowie ihre Ergebnis- und Benutzerorientierung machen es zudem notwendig, die Interessen und Einschränkungen, die ein Nutzer bei der Ausführung einer Aufgabe beachtet wissen will, auch in die Beschreibung Mobiler Prozesse zu integrieren. Diese *nicht-funktionalen Parameter* des Prozesses müssen also

ausdrückbar sein und während der Ausführung durch das ausführende (fremde) Gerät durchgesetzt werden (vgl. Abschnitt 5.4.5). Dabei sind die nicht-funktionalen Parameter nicht nur bei der Auswahl von passenden Dienstinstanzen zur Ausführung einzelner Aktivitäten notwendig, sondern auch bei der Auswahl der Migrationspartner, um Sicherheits- und Vertrauensforderungen des Nutzers einhalten zu können.

Insgesamt gilt also, dass durch die Migration der Kontrollfluss innerhalb einer mobilen Umgebung delegiert wird und so eine echte Kooperation zwischen den beteiligten Geräten erreicht werden kann. Hierbei bleibt die Autonomie des einzelnen Partners erhalten und doch wird die Ausführung der komplexen Aufgabe mobil. Erst die Migration zusammen mit einer Strategie zur späten Bindung von Dienstinstanzen ermöglicht die Ausführung von langlebigen Prozessen in einer mobilen und damit dynamischen Umgebung, in der Stabilitätsannahmen kaum gemacht werden können. Dabei muss bei der Ausführung und dem Transfer des Prozesses Kontextwissen berücksichtigt werden, um Dienste und Migrationspartner zu finden. Bei dieser Nutzung von Kontextdaten dienen nicht-funktionale Parameter als Rahmen, der die Absichten und Wünsche des Nutzers auch auf Geräten außerhalb seines Einflussbereichs definiert und somit die Ausführung entsprechend eingrenzt. Damit werden die Potenziale mehrerer Geräte einer mobilen Umgebung zum Vorteil und im Interesse des Prozessinitiators genutzt.

5.4.4. Erweiterung des Prozesslebenszyklus

Die Migrationseigenschaft von Mobilen Prozessen wirkt sich auch auf den Lebenszyklus, den ein Prozess während seiner Ausführung durchläuft (vgl. Abschnitt 4.4.1), aus. Dabei muss der Prozesslebenszyklus um neue interne Zustände erweitert werden, welche die Migration in die Navigation durch den Prozess einbinden (vgl. Abbildung 5.5).

Die Integration der Mobilitätseigenschaft wird durch das Einfügen eines neuen Zustands in den allgemeinen Prozesslebenszyklus erreicht, welcher der Ausführung der einzelnen Aktivität vorgelagert ist. Dieser sichere und als *Option* bezeichnete Zustand definiert einen Status, in dem der Prozess einen konsistenten Ausführungszustand erreicht hat und in dem keine Seiteneffekte durch die Ausführung weiterer Aktivitäten auftreten können. Er wird deshalb als Option bezeichnet, weil hier die Workflow Engine zwei Möglichkeiten hat, den Prozess voranzubringen. Dies ist zum einen die Möglichkeit, den Prozess selbst auszuführen und in den *Running*-Zustand

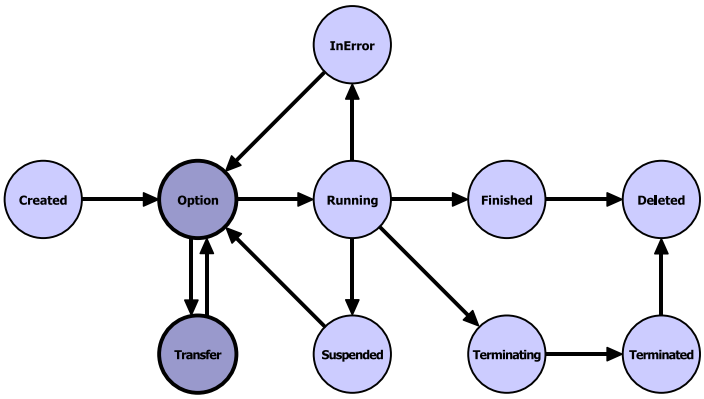


Abbildung 5.5.: Erweiterter Lebenszyklus für Mobile Prozesse

überzugehen. Zum anderen besteht aber auch die Gelegenheit, den Prozess an andere Geräte zu transferieren, indem in den *Transfer*-Zustand gewechselt wird. Der *Option*-Zustand gilt deshalb als sicher, da er nur dann erreicht werden kann, wenn entweder der Prozess gerade erzeugt, gezielt unterbrochen oder aber durch einen Fehler, wie zum Beispiel dem Fehlen einer Dienstinstanz zum Ausführen einer Aktivität, gestoppt wurde. Er beschreibt also einen Status der (temporären) Inaktivität, der es ermöglicht, den Anwendungszustand zu sichern und mit der Prozessbeschreibung zu übertragen. Der *Transfer*-Zustand stellt damit die Verbindung zwischen den Ausführungsumgebungen zweier Geräte dar: Auf dem einen Gerät verlässt der Prozess aus dem *Transfer*-Zustand heraus die lokale Workflow Engine, um auf einem anderen Gerät die neue Ausführungsumgebung in dem gleichen Zustand wieder zu betreten und weiter ausgeführt zu werden.

5.4.5. Integration nicht-funktionaler Aspekte zur Steuerung der Ausführung Mobiler Prozesse

Durch die Migration können Mobile Prozesse den Einflussbereich des sie initiierenden Benutzers verlassen. Aus diesem Grund gewinnen nicht-funktionale Parameter bei der Ausführung solcher Prozesse an Bedeutung. Denn erst durch sie ist es dem Nutzer möglich, seine Intentionen und Wünsche, die über die reine Aufgabe hinaus bestehen, auch auszudrücken. Sie bilden so einen Rahmen, der die Art und Weise der Ausführung steuert, auch wenn der Initiator nicht mehr direkt auf die Ausführung des Prozesses einwirken kann.

Nicht-funktionale Parameter sind dabei zunächst als *„Restriktionen über die Funktionalität eines Dienstes hinaus“* definiert [OEt02, Seite 118].

Diese Einschränkungen umfassen unter anderem zeitliche und örtliche Verfügbarkeiten, angebotene Kommunikationskanäle, Abrechnungsmodalitäten, Qualitätsparameter sowie Sicherheits- und Vertrauensaspekte. Dieser traditionelle Einsatzbereich von nicht-funktionalen Parametern muss jedoch für migrationsbasierte kooperative Ausführungsformen erweitert werden, da sich ihr Einsatz nicht auf die Dienstauswahl beschränkt, sondern Eigenschaften wie Sicherheit und Vertrauen auch bei der Wahl der Migrationspartner besonders wichtig sind. [OEt02]

Je nach Sensibilität der auszuführenden Aufgabe und der im Prozess enthaltenen Daten müssen unterschiedliche Restriktionsgrade bezüglich potenzieller Migrationspartner ausdrückbar sein. Diese reichen von keiner Einschränkung über die Einschränkung auf einzelne Geräte- beziehungsweise Benutzergruppen bis hin zur expliziten Vorgabe der Kommunikationspartner. Dabei werden Teile der benötigten nicht-funktionalen Parameter bereits durch etablierte Rollen- und Teilnehmerkonzepte von Prozessen und Workflows abgebildet, andere Aspekte hingegen – wie zum Beispiel das Benutzen von verschlüsselten Kommunikationskanälen oder der geforderte Vertrauensgrad eines Kommunikationspartners – müssen über erweiterte Konzepte integriert werden. Hierbei bietet sich ebenfalls ein generischer Ansatz an, da abermals die potenziell von einer Aufgabe geforderten Aspekte nicht umfassend im Voraus bestimmbar sind, sondern vielmehr der Ausführungsumgebung erst zur Laufzeit bekannt werden. Ungeachtet dessen ist die allgemeine Definition von gebräuchlichen nichtfunktionalen Parametern innerhalb des generischen Ansatzes sinnvoll, um eine einheitliche Semantik über Prozessgrenzen hinweg zu schaffen.

5.4.6. Sicherheit und Vertrauen als Basis kooperativen Verhaltens

Da bei der sich auf Migration stützenden kontextbasierten Kooperation das Delegieren von Prozessen an autonome und zum Teil unbekannte Geräte innerhalb einer mobilen Umgebung von zentraler Bedeutung ist, nimmt das Vertrauen, das sich die Migrationspartner entgegenbringen müssen, eine exponierte Stellung ein. Deshalb darf dieser Punkt bei der Konzeptualisierung migrierender Prozesse nicht unberücksichtigt bleiben. Und da Vertrauen nur in einem sicheren Umfeld entstehen kann, müssen auch entsprechende Sicherheitsaspekte mit berücksichtigt werden.

Aus diesem Grund ist es besonders wichtig, dass dezentrale Vertrauensmetriken, die eine quantitative Aussage über die wahrscheinliche Zuverlässigkeit eines Interaktionspartners machen, mit eingebunden werden

können. Hierbei ist aufgrund der möglichen Ad-hoc-Zusammenstellung einer mobilen Umgebung der Einsatz von Vertrauensmodellen, die auf zentrale Komponenten – zum Beispiel zum Verifizieren von digitalen Zertifikaten – setzen, oft nicht sinnvoll. Vielmehr sind Ansätze, die auf Empfehlungen und persönlichen Erlebnissen basieren (*Web of Trust*), beim Einsatz Mobiler Prozesse meist besser geeignet. Diese Vertrauenseigenschaften und grundlegenden Sicherheitsaspekte, wie zum Beispiel das Fordern von kryptografisch gesicherten Kommunikationsverbindungen, sind dabei als nicht-funktionale Attribute im Modell der Mobilen Prozesse mit berücksichtigt (siehe Abschnitt 5.4.5). [Zie05, EGB04]

5.4.7. Implikationen für ein Konzept zur Realisierung kontextbasierter Kooperation durch Mobile Prozesse

Um die kontextbasierte Kooperation durch verteilt ausgeführte Mobile Prozesse zu realisieren, müssen zunächst unterschiedliche Herausforderungen konzeptionell gelöst werden. Dabei sind die Anforderungen, die sich aus der neuen Klasse benutzerorientierter Middleware-Systeme im Mobile Computing (vgl. Tabelle 5.2), dem Einbeziehen komplexer Aufgaben in mobile Umgebungen (vgl. Tabelle 5.3) und der Realisierung als ein System mobiler, kooperierender und autonomer Geräte (vgl. Tabelle 5.4) ergeben, in die entsprechenden Konzepte einzubeziehen. Im Einzelnen müssen dabei Konzepte für folgende Bereiche erarbeitet werden:

- Zur Beschreibung komplexer Aufgaben als Mobile Prozesse muss ein *Prozessmetamodell* entwickelt werden, das die plattform- und programmiersprachenunabhängige Komposition von Teilaufgaben zu einem neuen Ganzen erlaubt. Dabei muss das Modell so gestaltet sein, dass zusätzlich zur abstrakten Beschreibung der Aufgabe auch der aktuelle Bearbeitungszustand eines Mobilen Prozesses sowie die zu seiner Ausführung notwendigen nicht-funktionalen Rahmenbedingungen definiert werden können und so die Migration der Aufgaben innerhalb der mobilen Umgebung ermöglicht wird (Abschnitt 5.5).
- Bei der Beschreibung Mobiler Prozesse müssen zudem die einzelnen *Teilaufgaben möglichst abstrakt* beschrieben werden können, sodass sie vollkommen unabhängig von den Diensten sind, die sie zur Laufzeit ausführen werden. Hierbei muss nicht nur von der einzelnen Dienstinstanz, sondern auch von den zur Realisierung der Instanzen verwendeten Technologien – wie Web Services oder der Remote Method Invocation –, abstrahiert werden, da im Allgemeinen zur

Entwurfszeit der Prozesse die Fähigkeiten der an der Ausführung beteiligten Geräte und die in ihrer Umgebung verfügbaren Dienstinstanzen nicht vorab bestimmt werden können (Abschnitt 5.5.4).

- Da die kooperative Ausführung von Prozessen in mobilen Umgebungen ohne zentrale administrative Komponenten auskommen können muss, sind *erweiterte Konzepte zur Definition und Ausführung von Transaktionen* für Mobile Prozesse notwendig (Abschnitt 5.7).

- Um nicht-funktionale Parameter bei der Wahl geeigneter Migrationspartner und Dienstinstanzen berücksichtigen zu können, muss ein *allgemeines und generisches Kontextkonzept* entwickelt werden, das mit a priori nicht bekannten Kontextinformationen umgehen kann (Abschnitt 5.6). Dabei muss vor allem auch ein Konzept eines technologieunabhängigen und verteilten Verzeichnisdienstes integriert werden, um kooperativ und damit möglichst effizient geeignete Dienstinstanzen zur Ausführung von Teilaufgaben in mobilen Umgebungen finden zu können. (Abschnitt 5.6.3).

- Abschließend muss ein *Realisierungskonzept* einer (prototypischen) Middleware für die Unterstützung Mobiler Prozesse erarbeitet werden, das die entwickelten Konzepte als integrierte Komponenten in einem Gesamtsystem zusammenfasst und so die kooperative Ausführung von Prozessen in mobilen Umgebungen ermöglicht (Kapitel 6).

Bei der Entwicklung der einzelnen Konzepte sind dabei stets die intrinsischen Eigenschaften mobiler Ausführungsumgebungen zu beachten, die damit einen axiomatischen Rahmen dieser Arbeit darstellen. Hierbei sollen alle Konzepte so entworfen sein, dass sie ein mobiles, ad-hoc gebildetes System gleichberechtigter und autonomer Teilnehmer als Ausführungsumfeld unterstützen. Deshalb ist, soweit möglich, auf den Einsatz von zentralen Komponenten zum Auslagern von Aufgaben oder ihrer Koordination zu verzichten.

5.5. Metamodell zur Beschreibung Mobiler Prozesse

Zur Realisierung kontextbasierter Kooperation ist zunächst ein abstraktes sowie plattform- und technologieunabhängiges Prozessmetamodell notwendig, das es erlaubt, komplexe Aufgaben als Mobilen Prozess zu beschreiben und durch die Integration des Prozesszustandes sowie von nichtfunktionalen Parametern zwischen unterschiedlichen Partnern einer mo-

bilen Umgebung auszutauschen. Dazu soll in diesem Abschnitt auf der Basis eines konsolidierten Anforderungskatalogs und einer Betrachtung der Eignung bestehender Modelle zur Beschreibung von Prozessen ein geeignetes Prozessmetamodell zur Definition von Mobilen Prozessen entwickelt werden.

5.5.1. Anforderungen an Metamodelle für Mobile Prozesse

Ein Metamodell zur Umsetzung der kontextbasierten Kooperation in Form von Mobilen Prozessen muss unterschiedliche Anforderungen der zuvor beschriebenen Konzepte umsetzen (vgl. Tabelle 5.5). Dabei muss ein Umsetzungsmodell zunächst in der Lage sein, *komplexe Aufgaben* zu beschreiben (MM1). Es muss also einzelne (elementare) Aufgaben zu einer neuen strukturierten Aufgabe zusammenfassen können. Dabei sollte es die aus der Modellierung von Geschäftsprozessen bekannten *Strukturen zur Abbildung von Daten- und Kontrollfluss* sowie zur Integration von spezifischen *Teilnehmern* und abstrakten *Rollen* aufgreifen und erweitern (MM2).

Im Zentrum dieser Erweiterung klassischer Prozesse hin zu Mobilen Prozessen steht dabei die Einführung von Elementen zur *Unterstützung der Migration* der Prozesse in mobilen Umgebungen (MM3). Dazu muss vor allem der Ausführungszustand des Prozesses gespeichert werden können, damit die Ergebnisse, die auf einem Gerät erzielt wurden auch auf anderen Geräten verfügbar sind. Des weiteren müssen *nicht-funktionale Parameter* definiert werden können, welche die Ausführung des Prozesses außerhalb der Einflusssphäre des Initiators seinen Anforderungen und Bedürfnisssen entsprechend steuern können (MM4).

Da a priori – also zur Entwurfszeit des Prozesses – durch die nichtdeterministische Migrationseigenschaft und die Heterogenität in mobilen Umgebungen nicht unbedingt bestimmt werden kann, welche technologischen Fähigkeiten das Gerät besitzen wird, das Teile des Prozesses tatsächlich ausführt, müssen die Teilaufgaben so abstrakt beschrieben werden, dass es erst zur Laufzeit möglich ist, zu entscheiden, welche Dienstinstanzen an die Aufgabe gebunden werden (*Spätes Binden*, MM5). Aus den gleichen Gründe ist es zudem notwendig, das Prozessmetamodell so zu gestalten, dass es *technologieunabhängig* ist (MM6). Es darf also keinen direkten Bezug zu Programmiersprachen, Betriebssystemen oder Kommunikationsprotokollen und -technologien aufweisen.

Da Mobile Prozesse sensible Informationen enthalten können und aus dem Einflussbereich des Initiators migrieren können, ist es notwendig,

Sicherheits- und Vertrauenseigenschaften von Kommunikationspartnern fordern zu können und so Einfluss auf die Auswahl der Migrationspartner und somit die den Mobilen Prozess ausführenden Geräte nehmen zu können (MM7). Diese Eigenschaften müssen dabei sowohl auf der Ebene einzelner Aktivitäten als auch auf der Ebene des Gesamtprozesses definiert werden können. Somit ist es dann möglich, unterschiedliche Grade an Sicherheit und Vertrauen zu definieren und diese an die Bedürfnisse des einzelnen Prozesses sowie des Prozessinitiators anzupassen.

Da Mobile Prozesse komplexe Prozesse abbilden können sollen, müssen auch *manuelle Aufgaben* in Form von *Benutzerinteraktionen*, die nicht rein computergesteuert abgearbeitet werden können, abbildbar und möglichst nahtlos in den Prozess integrierbar sein können (MM8). Hinzu kommt, dass durch lange Ausführungszeiten komplexer Prozesse die Einflüsse der unzuverlässigen Umgebung mobiler Systeme zunehmen. Deshalb ist es notwendig, *erweiterte Fehlerbehandungsmaßnahmen* in das Metamodell zu integrieren, da die Einflussnahme des Prozessinitiators durch die Migration in Fehlerfällen oftmals unmöglich ist (MM9).

Von besonderer Bedeutung bei der Ausführung von Prozessen und damit auch bei der Ausführung komplexer Aufgaben als Mobile Prozesse ist die Integration von *Transaktionen* (vgl. Abschnitt 5.7), um Teilaufgaben an eine gemeinschaftliche erfolgreiche Ausführung zu binden (MM10). Dabei werden wegen der Möglichkeit zur Migration und zur dezentralen Ausführung spezielle Transaktionskonzepte benötigt, die – soweit möglich – ein transaktionales Verhalten bei der Ausführung erlauben, ohne den Charakter der Mobilen Prozesse grundlegend zu verändern.

Schließlich muss ein Metamodell zur Beschreibung Mobiler Prozesse natürlich ebenfalls die *intrinsischen Eigenschaften mobiler Systeme* berücksichtigen (MM11). Dabei gilt es vor allem die Ausdrucksmöglichkeiten der Beschreibung so zu wählen, dass eine ressourcenschonende Interpretation, Ausführung und Übertragung möglich ist. Zudem müssen die schwankenden und unzuverlässigen Kommunikationsbeziehungen durch robuste und fehlertolerante Modellkonstrukte berücksichtigt werden.

Kürzel	Titel	Details
MM1	Komplexe Aufgaben	Mit Hilfe des Prozessmodells muss die Komposition einzelner elementarer Aufgaben zu komplexen Aufgaben möglich sein.

Fortsetzung von Tabelle 5.5 auf der nächsten Seite ...

Kürzel	Titel	Details
MM2	Prozessbasisstrukturen	Das Modell muss elementare Prozessstrukturen wie Daten- und Kontrollfluss sowie die beteiligten Rollen und Teilnehmer abbilden können.
MM3	Migration	Durch das Modell muss die Migration des Prozesses zwischen unterschiedlichen und heterogenen Geräten ermöglicht werden.
MM4	Nicht-funktionale Parameter	Da Mobile Prozesse den Einflussbereich ihres Initiators verlassen können, muss er durch das Einbinden von nicht-funktionalen Parametern seine Rahmenbedingungen zur Ausführung des Prozesses definieren können.
MM5	Spätes Binden / Dienstsuche zur Laufzeit	Durch die große Dynamik und Heterogenität in mobilen Umgebungen zusammen mit der Migrationseigenschaft von Mobilen Prozessen können zur Entwurfszeit keine Annahmen über die zur Laufzeit verfügbaren Dienstimplementierungen gemacht werden. Deshalb ist es notwendig, Dienste im Modell so zu definieren, dass sie erst zur Laufzeit gesucht und angesprochen werden können.
MM6	Technologieunabhängigkeit	Da Mobile Prozesse zwischen unterschiedlichen Ausführungsplattformen migrieren können, muss das Modell unabhängig von Programmiersprachen, Betriebssystemen und Netzwerkprotokollen sein.
MM7	Sicherheit / Vertrauen	Da Mobile Prozesse sensible Daten enthalten können, müssen zum gesamten Prozess und zu einzelnen Aktivitäten Sicherheits- und Vertrauensanforderungen definiert werden können, die während der Ausführung und Migration einzuhalten sind.

Fortsetzung von Tabelle 5.5 auf der nächsten Seite ...

Kürzel	Titel	Details
MM8	Benutzerinteraktion / manuelle Aufgaben	Da komplexe Aufgaben auch manuelle Tätigkeiten umfassen können, sollte es möglich sein, die Interaktion zwischen Prozess und ausführendem Akteur nahtlos in die Beschreibung einbinden zu können.
MM9	Fehlerbehandlung	Das Modell zur Beschreibung von Mobilen Prozessen muss es ermöglichen, das Verhalten in Fehlersituationen zu beschreiben. Dies ist notwendig, da durch die Migration des Prozesses ein direktes Einwirken des Prozessinitiators oftmals nicht machbar ist.
MM10	Transaktionen	Auch in Mobilen Prozessen sollte es möglich sein, Teile der enthaltenen Aktivitäten zu eng gekoppelten Einheiten zusammenzufassen und diese dann nur gemeinschaftlich erfolgreich auszuführen. Dazu muss das Modell das Definieren von Transaktionsgrenzen sowie die Ausführung unterschiedlicher Transaktionsformen bieten.
MM11	Intrinsische Eigenschaften mobiler Systeme	Da Mobile Prozesse auch auf mobilen Geräten ausgeführt werden, muss das Modell die intrinsischen Eigenschaften mobiler Systeme berücksichtigen. Dabei müssen vor allem die Prozesse möglichst ressourcensparend verarbeitet und ausgeführt werden können sowie robust gegenüber schwankenden und unzuverlässigen Kommunikationsbeziehungen sein.

Tabelle 5.5.: Konsolidierte Anforderungen an ein Prozessmetamodell (nach (ZK07, KZL07b))

5.5.2. Defizite bestehender Prozessbeschreibungssprachen

Wie in Kapitel 4 beschrieben eignen sich Prozessbeschreibungssprachen besonders gut, um komplexe Aufgaben abstrakt zu definieren und aus-

zuführen. So existieren bereits unterschiedliche Sprachen, mit deren Hilfe Prozesse beschrieben werden können. In welchem Umfang diese Sprachen auch zur Umsetzung des Konzeptes Mobiler Prozesse eingesetzt werden können, wird an dieser Stelle mit Hilfe des zuvor aufgestellen Anforderungskataloges ermittelt. Hierzu werden prominente Prozessmodellierungssprachen wie *WS-BEPEL*, *WSCI* (dem Vorgängerstandard von *WSCDL*) und *XPDL* (vgl. Abschnitt 4.5.1 und 4.5.3) sowie die Modellierungssprachen *jBPM Process Definition Language* (JPDL) [JBo07] und das *Business Process Specification Schema* des EbXML-Rahmenwerks (EbBPSS) [CCK+01] auf ihre Eignung bezüglich der ermittelten Anforderungen hin untersucht (vgl. Tabelle 5.6). [ZK07]

Für alle untersuchten Prozessbeschreibungssprachen gilt dabei, dass sie, besonders auch wegen ihrer Erweiterbarkeit, für die Abbildung komplexer Aufgaben grundsätzlich geeignet sind (MM1) und hierbei ausreichende Fähigkeiten zur Definition grundlegender Prozessstrukturen – wie Daten- und Kontrollfluss sowie Rollen und Teilnehmer – aufweisen (MM2). Zugleich gilt jedoch auch, dass alle untersuchten Prozessbeschreibungssprachen den Ansatz der kontext-basierten Kooperation und damit der Migration der Prozesse zwischen unterschiedlichen Ausführungseinheiten nicht direkt unterstützen (MM3). *Damit gilt, dass keine der Prozessbeschreibungssprachen ohne Einschränkungen und Ergänzungen zur Abbildung Mobiler Prozesse geeignet ist.* Deshalb kann das Ergebnis dieser Untersuchung nur sein, eine existierende Sprache zu finden, die eine Grundlage für eine erweiterte Prozessmodellierungssprache zur Beschreibung Mobiler Prozesse darstellt.

Die im Umfeld Mobiler Prozesse zur Auswahl von Diensten, deren Ausführung sowie zur Bestimmung von Migrationspartnern besonders wichtigen nicht-funktinalen Parameter (MM4) werden von *WSCI* und *EbBPSS* bereits nativ unterstützt. *WS-BEPEL*, *JPDL* und *XPDL* hingegen unterstützen die Definition und Nutzung in den untersuchten Versionen nicht unmittelbar – können aber um diese ergänzt werden.

Das späte Binden von Dienstinstanzen (MM5) wird von allen betrachteten Ansätzen zumindest ansatzweise unterstützt. Dabei bieten *WSCI* und *EbBPSS* jedoch die schwächste Unterstützung. So muss bei *WSCI* für jede Aktivität bereits im Vorfeld der Ausführung eine Menge von potenziellen Dienstinstanzen definiert werden, aus der dann zur Laufzeit eine Instanz bestimmt werden kann, die dann tatsächlich die Aktivität ausführt. Eine solche a priori Festlegung auf eine Menge von Dienstinstanzen ist jedoch im Umfeld Mobiler Prozesse meist nicht möglich. *EbBPSS*

ist hauptsächlich auf die Abbildung binärer Geschäftsbeziehungen ausgelegt und bietet keine einfache Auswahlmöglichkeit von gleichwertigen Dienstinstanzen für eine Aktivität. Es ist jedoch möglich, über den Umweg von *Multi Party Collaborations* und entsprechenden einschränkenden Eigenschaften, eine späte Bindung nachzubilden. *WS-BEPEL*, *JPDL* und *XPDL* ermöglichen hingegen ein spätes Binden relativ komfortabel, da es möglich ist, durch die Trennung von Dienstbeschreibung und ausführendem Teilnehmer eine Abstraktion von einzelnen Ausführungsinstanzen zu erreichen.

Aufgrund der heterogenen Struktur mobiler Umgebungen und damit der Geräte, die Mobile Prozesse ausführen, ist die von Technologien und Kommunikationsprotokollen unabhängige Beschreibung von Mobilen Prozessen besonders wichtig (MM6). Hierbei bietet *JPDL* die schwächste Umsetzung, da sie auf die JAVA-Programmiersprache und Ausführungsumgebung angewiesen ist. *WS-BEPEL* und *WSCI* sind als Erweiterungen des Web-Service-Protokollstapels auf dessen Nutzung beschränkt und bieten somit lediglich eine Unabhängigkeit von den Umsetzungstechnologien der Web Services. *XPDL* bietet als Metasprache zum Austausch von Prozessbeschreibungen inhärent eine Abstraktion von konkreten Dienstinstanzen und Umsetzungstechnologien.

Die Integration von Sicherheits- und Vertrauensmechanismen (MM7) wird von den untersuchten Ansätzen kaum unterstützt. Lediglich *EbBPSS* bietet aufgrund seiner geschäftlichen Ausrichtung vorbestimmte Attribute an, die zur Etablierung gewisser Vertrauensgrade führen kann. So ist es zum Beispiel möglich, einzelne Transaktionen als *„bindend“* zu kennzeichnen, um zu bestimmen, dass durch diese Transaktion ein rechtsverbindlicher Vertrag geschlossen werden soll.

Durch die Ausrichtung auf die computergestützte Ausführung von Geschäftsprozessen bieten *WS-BEPEL*, *WSCI* und *EbBPSS* keine direkte Integration des Benutzers in Form von manuellen Aktivitäten (MM8) an. Im Gegensatz hierzu können bei *JPDL* und *XPDL* auch manuelle Aktivitäten direkt in den Prozess einbezogen werden.

In unzuverlässigen Umgebungen wie in denen mobiler Systeme ist eine adäquate Fehlerbehandlung von besonderer Bedeutung (MM9). Die Beschreibung von expliziten Fehlerbehandlungsmaßnahmen ist jedoch lediglich in den Sprachen *WS-BEPEL* und *WSCI* möglich. Rudimentäre, aber nicht ausreichende Ansätze zur prozessspezifischen Fehlerbehandlung finden sich noch bei *JPDL* und *EbBPSS*. *XPDL* bietet hierbei die schlechteste Unterstützung von den betrachteten Ansätzen.

Transaktionales Verhalten (MM10) wird ebenfalls nur von den Sprachen *WS-BEPEL* und *WSCI* unterstützt. *JPDL*, *EbBPSS* und *XPDL* bieten hingegen keine Ausdrucksmöglichkeiten für Transaktionen in ihren Beschreibungssprachen an.

Da keine der etablierten und untersuchten Prozessbeschreibungssprachen explizit für mobile Umgebungen entworfen wurde, unterstützen diese auch nicht uneingeschränkt die intrinsischen Eigenschaften mobiler Systeme. Jedoch haben *WSCI*, *JPDL* und *XPDL* durch eine kompakte Beschreibungssprache, eher moderate Ansprüche an die Rechenleistung ausführender Systeme. Zudem bieten ihre Ansätze zur lokalen Datenhaltung einen Vorteil, der auch von einer Prozessbeschreibungssprache für mobile Umgebungen geboten werden sollte.

Insgesamt hat sich bei der Analyse existierender Modellierungssprachen die Metabeschreibungssprache *XPDL* als gute Grundlage für eine Sprache zur Definition Mobiler Prozesse herausgestellt. Sie bietet die Möglichkeit, komplexe Aufgaben als Prozesse zu definieren und ist dabei vollkommen plattform-, sprach- und protokollunabhängig. Zudem erlaubt sie durch ihre abstrakte Definition der Aktivitäten eine späte Bindung und kann auch manuelle Aktivitäten integrieren. Durch ihre Erweiterbarkeit können zudem die fehlende Unterstützung von nicht-funktionalen Parametern sowie Konstrukte zur Unterstützung der Migration und erweiterten Fehlerbehandlung eingefügt werden. Durch zusätzliche vereinfachende Sprachkonstrukte, zum Beispiel für die einfache Definition von Schleifen, kann *XPDL* zudem weiter auf die Bedürfnisse mobiler Systeme zugeschnitten werden. Auch die fehlende Unterstützung von Transaktionen kann ebenfalls nachträglich durch weitere Ergänzungen integriert werden.

Kürzel	Titel	WS-BEPEL	WSCI	EbBPSS	JPDL	XPDL
MM1	Komplexe Aufgaben	+	+	+	+	+
MM2	Prozessbasisstrukturen	+	+	+	+	+
MM3	Migration	−	−	−	−	−
MM4	Nicht-funktionale Parameter	−	+	+	−	−
MM5	Spätes Binden / Dienstsuche zur Laufzeit	+	◯	◯	+	+

+ = unterstützt − = nicht unterstützt ◯ = teilweise unterstützt

Fortsetzung von Tabelle 5.6 auf der nächsten Seite . . .

Kürzel	Titel	WS-BEPEL	WSCI	EbBPSS	JPDL	XPDL
MM6	Technologieunabhängigkeit	○	○	○	−	+
MM7	Sicherheit / Vertrauen	−	−	+	−	−
MM8	Benutzerinteraktion / manuelle Aufgaben	−	−	−	+	+
MM9	Fehlerbehandlung	+	+	−	−	−
MM10	Transaktionen	+	+	−	−	−
MM11	Intrinsische Eigenschaften mobiler Systeme	−	○	−	○	○

+ = unterstützt − = nicht unterstützt ○ = teilweise unterstützt

Tabelle 5.6.: Bewertung bestehender Prozessmetamodelle (nach (ZK07, KZL07b))

5.5.3. Konzeptionelles Prozessmodell

Das konzeptionelle Modell zur Realisierung Mobiler Prozesse besteht aus einer Prozessbeschreibungssprache, die es erlaubt, komplexe Aufgaben, die nicht-funktionalen Rahmenbedingungen der Anwendung und des Prozessinitiators zu definieren. Mit Hilfe der Sprache können Sequenzen von Aktivitäten, gewünschte Zwischen- und Endergebnisse und die einschränkenden Bedingungen beschrieben werden. Sie ist dabei durch ein XML-Schema[2] definiert und basiert auf der Sprache *XPDL Version 1.0*. Dabei erbt sie von ihr die Sprachkonstrukte, die sich während der Analyse existierender Sprachen als zweckmäßig erwiesen haben (vgl. Tabelle 5.6). [ZK07, KZL07b]

5.5.3.1. Aufbau des Metamodells

Das Basiselement des Prozessmodells zur Beschreibung Mobiler Prozesse ist – wie in *XPDL* – ein *Package*, das einen Container für alle den Prozess beschreibenden Elemente und die zu seiner Ausführung relevanten Daten (*Workflow Relevant Data*) darstellt (vgl. Abbildung 5.6). Hierbei wird jedoch gegenüber *XPDL* die Anzahl der enthaltenen Prozesse (*WorkflowProcess*) auf einen einzigen beschränkt, um den Ressourcenrestriktionen mobiler Geräte gerecht zu werden und die Prozesse einzeln zwischen

[2] http://vsis-www.informatik.uni-hamburg.de/projects/demac/dpdl1.1.xsd

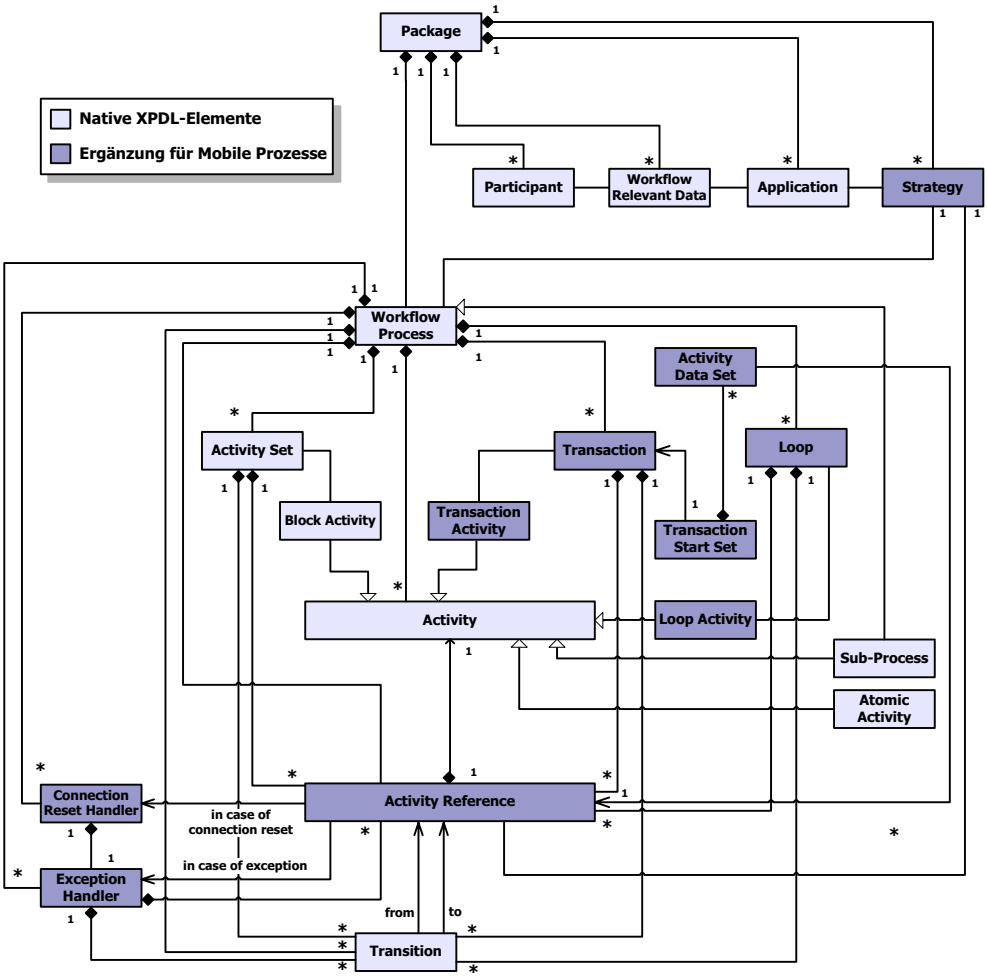


Abbildung 5.6.: Prozessmodell zur Beschreibung Mobiler Prozesse (nach (KZL07b, KZL06))

unterschiedlichen Migrationspartnern austauschen zu können. Neben dem Prozess werden auch weitere wiederverwendbare beziehungsweise global gültige Elemente – wie spezifische Teilnehmer (*Participants*), Anwendungen in Form von abstrakten Diensten (*Applications*) und speziell für Mobile Prozesse nicht-funktionale Bedingungen als Strategien (*Strategy*) – durch entsprechende Subelemente des *Packages* definiert. [ZK07, KZL07b]

Im *Workflow Process* ist der Kontrollfluss einer komplexen Aufgabe durch die Angabe ihrer Aktivitäten (*Activity)* und den zwischen ihnen bestehenden Transitionen (*Transition*) beschrieben. Dabei können Aktivitäten entweder atomar sein, selbst wieder einen Subprozess repräsentieren oder als *Block Activity* eine wiederverwendbare Sequenz von Aktivitäten (*Activity Set*) vertreten. Zusätzlich können Aktivitäten in Form von *Transaction Activities* zu Transaktionen zusammengefasst werden.

Durch das Einführen von expliziten Schleifenelementen (*Loop Activity*) kann die Beschreibungssprache für Mobile Prozesse wiederholbare Prozessabschnitte effizienter beschreiben, als es in *XPDL* möglich ist und so den aufgestellten Anforderungen (vgl. Abschnitt 5.5.2) gerechter werden. Dabei werden die Aktivitäten durch die Verwendung von *Activity References* wiederverwendbar. Diese Referenzen sind innerhalb des Prozesses eindeutig und enthalten alle Informationen, die in Bezug auf die Position einer Aktivität innerhalb des Kontrollflusses variabel sind. Dies können spezielle Teilnehmer, nicht-funktionale Bedingungen oder Fehlerbehandlungsmaßnahmen sein. Zusätzlich bilden sie die Grundlage des Kontrollflusses, da Transitionen jeweils zwei dieser Referenzen zu einem Kontrollflussabschnitt zusammenfügen. [ZK07, KZL07b]

Nicht-funktionale Bedingungen, wie zum Beispiel geforderte Qualitätskriterien, können als Strategien nicht nur durch Subelemente des *Packages* definiert werden. Es ist vielmehr auch möglich, Strategien, die sich nur auf einzelne Aktivitäten und damit auch auf Subprozesse, Transaktion oder Schleifen beziehen, diesen einzeln zuzuordnen. [ZK07, KZL07b]

Um den unzuverlässigen Rahmenbedingungen mobiler Umgebungen gerechter zu werden, sind *Connection Reset Handler* und *Exception Handler* als zusätzliche Elemente in die Beschreibungssprache aufgenommmen worden. Diese referenzieren wiederum eine Sequenz von Aktivitäten, die ausgeführt werden soll, falls entsprechende Fehlerfälle auftreten. Damit können auch in Ausnahmesituationen, die von der eigentlichen Beschreibung der komplexen Aufgabe abweichen, Maßnahmen definiert werden, um zum Beispiel den Initiator eines Prozesses über das endgültige Scheitern des Prozesses per E-mail zu informieren. [ZK07, KZL07b]

5.5.3.2. Zustandskonzept

Um die opportunistische Ausführungsstrategie (vgl. Abschnitt 5.4.1) und damit die Migration Mobiler Prozesse durch die Beschreibungssprache unterstützen zu können, muss der Zustand des Prozesses in der ausgetauschten Beschreibung festgehalten werden können. Nur so ist es möglich den Prozess anzuhalten, ihn an andere Geräte in der mobilen Umgebung zu übertragen und dort wieder zu starten, ohne dass bereits erzielte Ergebnisse verloren gehen. Dabei besteht der Gesamtzustand des Mobilen Prozesses nicht nur aus den bis dahin erzielten Daten, die in den Variablen des Prozesses gespeichert sind, sondern auch in den Ausführungszuständen der einzelnen Aktivitäten. Diese lokalen Subzustände werden dabei durch

den Lebenszyklus für Aktivitäten (vgl. Abschnitt 4.4.1) bestimmt und jeweils als Eigenschaft an die einzelne *Activity Reference* gebunden. Damit ist der Gesamtzustand des Prozesses während seiner Ausführung stets eindeutig definiert und für jedes an der Ausführung beteiligte Gerät sichtbar.

Zu Beginn der Ausführung eines Mobilen Prozesses befinden sich alle seine *Activity References* im Zustand *Inactive* und bleiben in diesem Zustand, solange im Kontrollfluss vorgelagerte Aktivitäten nicht ausgeführt wurden oder Vorbedingungen für die Ausführung der Aktivität noch nicht erfolgreich ausgewertet wurden. Sollten solche Vorbedingungen endgültig nicht mehr erfüllbar sein, geht die *Activity Reference* in den Fehlerzustand *Skipped* über. Ansonsten wird nach dem erfolgreichen Auswerten der Vorbedingungen der Zustand *Ready* eingenommen und die Aktivität kann potenziell ausgeführt werden. Führt ein Gerät eine potenziell ausführbare Aktivität nicht aus, sondern migriert stattdessen den Prozess, müssen sämtliche Vorbedingungen erneut überprüft werden, da sich die Ausführungsumgebung geändert hat und somit zum Beispiel zeitliche Restriktionen oder geforderte Fähigkeiten nicht mehr gültig sind. Sollte eine zeitliche Vorgabe zur Ausführung nicht mehr gültig sein, wird die Aktivität mit dem Zustand *Expired* gekennzeichnet. Das Auftreten von *Skipped*- und *Expired*-Zuständen kann dazu führen, dass nachgelagerte Aktivitäten nicht mehr zur Ausführung kommen können. Deshalb müssen zum Erkennen des Gesamtzustandes nicht-ausführbare Prozesszweige durch eine *Dead Path Elimination* erkannt und entsprechend gekennzeichnet werden. [ZK07, KZL07b]

Befindet sich eine *Activity Reference* im Zustand *Ready* kann sie ausgeführt werden und geht dazu in den Status *Executing* über. Während der lokalen oder entfernten Ausführung einer Aktivität wird der Gesamtprozess nicht migriert, vielmehr muss abgewartet werden, ob der Dienstaufruf erfolgreich oder fehlerhaft ist. Ist die Aktivität erfolgreich ausgeführt, nimmt sie den Zustand *Executed* an und kann entweder als endgültig bearbeitet gekennzeichnet werden (*Finished*) oder aber, zum Beispiel als Teil einer Schleife, wieder in den *Ready*-Zustand versetzt werden. [ZK07, KZL07b]

Um den Start der Ausführung eines Mobilen Prozesses nach einer Migration zu beschleunigen, wird zusätzlich die als nächstes auszuführende Aktivität als *Start Aktivität* definiert. Damit wird vermieden, dass ein Gerät sich diese Aktivität erst durch Auswerten der Aktivitätszustände und der entsprechenden Aktivierungsbedingungen ableiten muss. [ZK07, KZL07b]

5.5.3.3. Integration externer Daten

Um besonders effizient mit den Ressourcen mobiler Geräte umzugehen, ist es in der Beschreibungssprache für Mobile Prozesse möglich, Daten zu externalisieren. Dabei werden sie als Link zur Datenquelle beschrieben und müssen bei Bedarf von der Ausführungsumgebung nachgeladen werden. Bei der Verwendung dieses Mechanismus muss jedoch zwischen der Reduktion des Umfangs der Beschreibung und der Verfügbarkeit der Datenquelle abgewogen werden. Deshalb sollte die Auslagerung von Daten nur dann eingesetzt werden, wenn es von der Anwendung her sinnvoll erscheint. So ist es zum Beispiel denkbar, dass eine Aktivität von einem bestimmten Teilnehmer ausgeführt werden soll und dieser die Verfügbarkeit der zur Ausführung notwendigen Daten garantieren kann, sodass in diesem Fall die Auslagerung der Daten sinnvoll sein kann. [ZK07, KZL07b]

5.5.3.4. Einbinden von Teilnehmern und Geräten

Da die Geräte in mobilen Umgebungen häufig direkt einem Nutzer zugeordnet sind, ist es sinnvoll und nützlich, diesen in die Ausführung von Mobilen Prozessen einbeziehen zu können. Eine solche Integration kann von manuellen Aktivitäten innerhalb der komplexen Aufgabe bis zur Fehlerbehandlung in Ausnahmesituationen reichen. Hierbei können entweder gezielt Personen, zum Beispiel mit Hilfe von *digitalen Identitäten*, adressiert werden oder es werden abstrakte Nutzergruppen über ihre *Rolle* angesprochen. Dabei können auch einzelne Geräte über eindeutige Bezeichner, wie zum Beispiel eine MAC-Adresse, als Ausführungsort einer Aktivität bestimmt werden. In Kombination kann man so sehr gezielt steuern wie sich ein Mobiler Prozess verhalten soll. Dies ist vor allem dann notwendig, wenn zum Beispiel Ergebnisse des Prozesses an einen Nutzer übermittelt werden sollen. [ZK07, KZL07b]

5.5.3.5. Fehlerbehandlung und Verbindungsabbrüche

Da mobile Umgebungen wesentlich fehleranfälliger sind als stationäre sowie drahtgebundene Systeme und Mobile Prozesse zudem den Verantwortungsbereich des Initiators verlassen können, ist die Integration von gezielten Fehlerbehandlungsmaßnahmen in die Prozessbeschreibung essenziell. Dabei sollten durch die Fehlerbehandlung möglichst alle Informationen, die zum Beheben des Fehlers oder eines unvorhergesehenen Verbindungsabbruches notwendig sind, bereitgestellt werden. Zumindest sollte ein endgültiges Scheitern eines Mobilen Prozesses, dessen Initiator etwa

in Form einer E-Mail oder SMS bekannt gegeben werden, damit dieser entsprechend reagieren kann. [ZK07, KZL07b]

Da eine adäquate Fehlerbehandlung jedoch meistens nur in Abhängigkeit zum einzelnen Prozess durchgeführt werden kann, muss sie in der Prozessbeschreibung ausdrückbar sein und dem Modellierer eine größtmögliche Flexibilität bieten. Dies wird in der Beschreibungssprache für Mobile Prozesse durch *Exception Handler* und *Connection Reset Handler* erreicht, die mit Bedingungen versehen werden können und die beim Eintreten entsprechender Ereignisse einen alternativen Kontrollfluss anstoßen. Damit steht dem Modellierer prinzipiell die gesamte Ausdrucksmächtigkeit der Mobilen Prozesse zur Fehlerbehandlung zur Verfügung. [ZK07, KZL07b]

In Situationen, in denen Geräte dauerhaft ausfallen, scheitern natürlich auch die von ihnen ausgeführten Prozesse. Um auch in solchen Situationen einen Prozess nicht endgültig scheitern zu lassen, kann durch die Integration von *Check Points* und dem Halten von Kopien der Prozessbeschreibungssprache an unterschiedlichen Orten auch in einem solchen pessimistischen Szenario eine Fehlerbehandlung erreicht werden. [ZK07, KZL07b]

5.5.4. Abstraktion durch das Bilden von Dienstklassen

Bei der Beschreibung von Aktivitäten für Mobile Prozesse muss darauf geachtet werden, dass eine möglichst technologieunabhängige Integration von heterogenen Anwendungen und Diensten erreicht wird, da zur Entwurfszeit der Prozesse in der Regel noch nicht bekannt ist, welche Arten von Diensten und Anwendungen im Umfeld des ausführenden (mobilen) Gerätes vorhanden sind. Deshalb muss eine abstrakte Beschreibung der Aktivitäten gefunden werden, die das späte Binden von a priori unbekannten Dienstinstanzen und somit auch von Implementierungstechnologien zur Laufzeit ermöglicht. [KZL07a]

Das für die kontextbasierte Kooperation entwickelte Konzept der *abstrakten Dienstklassen* greift das von Middleware-Plattformen zum Dienstaufruf in verteilten Umgebungen bekannte Konzept der abstrakten (aber technologieabhängigen) Schnittstellenbeschreibung auf und führt durch die konsequente Trennung von Implementierungstechnologie und Schnittstelle eine weitere Abstraktionsstufe zur Schnittstellenbeschreibung ein. Es umfasst demnach nur die Metadaten, die allen Implementierungen gemein sind und zum eindeutigen Auffinden der (Teil-)Aufgaben in Form von Diensten benötigt werden. Allgemein können *abstrakte Dienstklassen* dem-

nach folgendermaßen definiert werden:

> *Eine abstrakte Dienstklasse subsumiert alle technologieabhängigen Beschreibungen eines Dienstes unter einer gemeinsamen und technologieunabhängigen Referenz. Sie wird durch eine Signatur definiert, die aus einem eindeutigen Bezeichner und einer Parameterliste besteht.*
>
> [KZL07a, Seite 125]

Abstrakte Dienstklassen fassen gleichartige Dienstinstanzen durch eine Abstraktion auf drei Ebenen zusammen, damit diese für die kontextbasierte Kooperation mit Mobilen Prozessen genutzt werden können (vgl. Abbildung 5.7): Die unterste Ebene bilden die *instanzabhängigen Beschreibungen* der einzelnen Dienstimplementierungen. Sie beschreiben die Charakteristika und Referenzen, die notwendig sind, um eine spezifische Dienstinstanz anzusprechen. Bei der Realisierung eines Dienstes als *Web Service* wären dies zum Beispiel die *Binding*-Informationen. Die Ebene darüber wird durch die *technologieabhängigen abstrakten Beschreibungen* gebildet, welche die Schnittstelleninformationen der Dienste in allgemeiner, jedoch für die jeweilige Middleware-Technologie spezifischer Form enthält. In einer Umgebung, die *CORBA* zum Aufruf verteilter Dienste nutzt, wäre dies zum Beispiel die Beschreibung der Schnittstelle als IDL-Dokument. Die

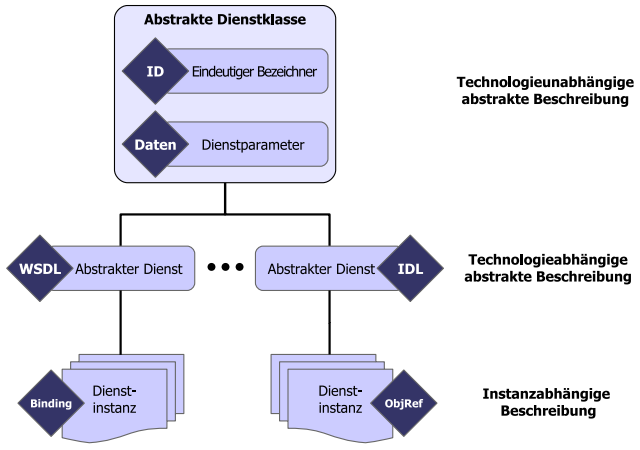


Abbildung 5.7.: Hierarchie abstrakter Dienstklassen

oberste Ebene der *abstrakten Dienstklassen* beziehungsweise der *abstrakten und technologieunabhängigen Beschreibungen* fasst nun wiederum die Gemeinsamkeiten der darunterliegenden Schicht in einer einheitlichen Beschreibung zusammen. Hierbei wird sowohl ein eindeutiger Bezeichner für den Dienst vergeben, als auch die Liste der zum Aufruf notwendigen Para-

meter mit einbezogen. Dies ist notwendig, da zur Beschreibung des Datenflusses eines (Mobilen) Prozesses angegeben werden muss, welche Variablen des Prozesses an welche Parameter der Dienstklasse gebunden werden müssen. [KZL07a]

Zum Auflösen der Dienstklassen zur Laufzeit muss dann ebenfalls ein mehrstufiges Verfahren angewendet werden. In diesem werden auf Basis des Kontextwissens (vgl. Abschnitt 5.6) zunächst die vom (mobilen) Gerät beherrschten Technologien als zusätzliche nicht-funktionale Parameter ermittelt und mit Hilfe einer verteilten Registratur (vgl. Abschnitt 5.6.3) in ausführbare Dienstobjekte überführt. Diese sind als Wrapper-Objekte konzipiert, die eine generische Schnittstelle zur Verfügung stellen, mit deren Hilfe unterschiedliche Dienstimplementierungen einheitlich angesprochen werden können.

5.6. Kontext als ergänzende Wissensbasis zur Ausführung Mobiler Prozesse

Damit Mobile Prozesse nicht nur ihre eigenen Fähigkeiten und Ressourcen nutzen können, sondern auch die der gesamten mobilen Umgebung, migrieren sie zwischen den sich hierin befindenden Geräten. Dabei werden beim Migrieren und der Ausführung einzelner Aktivitäten funktionale und nicht-funktionale Anforderungen berücksichtigt, um Dienste, Teilnehmer und Geräte zur Interaktion zu bestimmen. Da die hierbei benötigten Attribute jedoch vom einzelnen Prozess abhängen und damit vorab nicht allgemein bestimmt werden können, ist ein spezifisches Kontextmodell ungeeignet, Mobile Prozesse zu unterstützen. Vielmehr wird ein allgemeines, flexibles und generisches Kontextmodell benötigt, das Auskunft über die Geräte, Dienste und Parameter des aktuellen Ausführungskontextes des Mobilen Prozesses gibt.

5.6.1. Anforderungen an ein Kontextmodell für Mobile Prozesse

Ein *Kontextmodell* zur Unterstützung der Ausführung Mobiler Prozesse muss, da erst zur Laufzeit auf dem aktuellen Gerät bestimmt werden kann, welche Kontextinformationen zur Ausführung eines einzelnen Prozesses notwendig sind, möglichst *allgemein* und *generisch* gehalten werden (KM1) und bei Bedarf *flexibel erweitert* werden können (KM2). Dabei müssen die jeweils *relevanten Kontextdaten* der mobilen Umgebung auf passende Datenstrukturen abgebildet werden und standardisiert zugreifbar sein. Um hierbei Wiederverwendbarkeit und Erweiterbarkeit sicherzustellen, sollte

das *Kontextmodell konzeptionell von der Managementkomponente entkoppelt* werden (KM3) und dabei *unabhängig von Plattformen, Kommunikationsprotokollen sowie Programmiersprachen* sein. Durch diese Abstraktion kann die in mobilen Systemen oftmals bestehende Heterogenität überwunden werden und das Kontextkonzept in bestehende (mobile) Systeme integriert werden (KM4). [KZTL08, Tur06]

Da Mobile Prozesse durch ihre Migrationsfähigkeit eine Kooperation unterschiedlicher Geräte ermöglichen, werden neben dem lokalen Kontext auch die Kontextdaten anderer für die Ausführung der Prozesse wichtig. Aus diesem Grund müssen Kontextinformationen zwischen den Teilnehmern ausgetauscht und zu einem erweiterten verteilten Kontext vereinigt werden (KM5). Hierzu muss durch ein *Föderationskonzept* die Möglichkeit geschaffen werden, Kontextdaten mit Hilfe *standardisierter Protokolle* auszutauschen, um so einen Zugriff auf die Kontexte kooperierender Geräte zu ermöglichen und einen globalen Kontext aufzubauen. Hierbei sollten aufgrund der Dynamik mobiler Umgebungen möglichst nur lokale Daten permanent gespeichert werden, die dann bei Bedarf durch Daten entfernter Geräte ergänzt werden. Um dabei von unterschiedlichen Repräsentationen einzelner Kontextdaten unabhängig zu sein, müssen *abstrakte Repräsentationsformen* gewählt werden, die neben den reinen Kontextattributen auch *semantische Metainformationen* bieten (KM6). Diese zusätzlichen Angaben können dann dazu genutzt werden, Informationen in äquivalente lokale Repräsentationen zu transformieren (KM7). Hierbei ist es sinnvoll, Daten anhand ihrer unterschiedlichen Änderungsraten (statisch, dynamisch) und ihrem jeweiligen Abstraktionsgrad (high-level, low-level) zu unterscheiden (KM12, vgl. Abschnitt 3.3). Um hierbei automatisch über Änderungen dynamischer Kontextdaten auf entfernten Geräten informiert zu werden, sollten die Protokolle zum Bilden der Kontextföderation auch *proaktive Kommunikationsformen* – wie zum Beispiel die Publish/Subscribe-Kommunikation (vgl. Abschnitt 3.2.2) – integrieren (KM8). [KZTL08, Tur06]

Auch beim Kontextmodell müssen natürlich die intrinsischen Eigenschaften mobiler Systeme (vgl. Abschnitt 2.3.2) berücksichtigt werden. Deshalb muss das Kontextkonzept und die zur Verwaltung föderierter Kontextdaten benötigten Protokolle und Komponenten möglichst *leichtgewichtig* und zudem *robust* gegenüber Fehlern sein, die sich aus der unzuverlässigen mobilen Umgebung ergeben (KM9). Dies bedeutet im Besonderen, dass die Dienste und Modellelemente so entworfen sind, dass sie *flexibel* und damit *skalierbar* eingesetzt werden können, um zum Beispiel

nicht benötigte Komponenten zeitweise deaktivieren zu können und so die Ressourcen der (mobilen) Geräte zu schonen (KM10). Dabei sollten nur relevante Daten ausgetauscht werden, damit die zur Verfügung stehenden und meist unzuverlässigen Kommunikationsmöglichkeiten möglichst effizient genutzt werden. Deshalb ist es notwendig, adäquate *Filtermechanismen* zu integrieren, um situationsspezifische Daten und somit jeweils den für den einzelnen Prozess relevanten Kontextausschnitt definieren zu können (KM11). [KZTL08, Tur06]

Darüber hinaus müssen auch beim Umgang mit Kontextdaten Aspekte von *Sicherheit* und *Vertrauen* berücksichtigt werden (KM13). Denn auch Kontextattribute können sensible Daten über einen Nutzer preisgeben, sodass Sichtbarkeitskonzepte zum Einschränken des Zugriffs auf lokale Kontextinformationen sowie sichere Kommunikations- und Authentifizierungsmethoden berücksichtigt werden müssen. Zudem sind auch Angaben zur *Qualität von Kontextdaten* ein wichtiges Kriterium, um die Güte der bereitgestellten Information zu bewerten (KM14). So kann zum Beispiel eine Angabe zur aktuellen Position eines Gerätes präzise durch einen eingebauten GPS-Empfänger ermittelt worden sein oder nur durch eine Approximation mittels einer Peilung innerhalb einer Mobilfunkzelle. Betrachtet man beide Ergebnisse als gleichwertig, könnte sich dies auf die Qualität der von einer Anwendung erbrachten Leistung negativ auswirken und im Extremfall sogar zu Fehlfunktionen führen. [KZTL08, Tur06]

Kürzel	Titel	Details
KM1	Bereitstellen von Kontextdaten	Kontextdaten des lokalen Gerätes und der mobilen Umgebung müssen einheitlich bereitgestellt werden, um a priori nicht bekannte (komplexe) Anwendungen unterstützen zu können.
KM2	Erweiterbarkeit und Wiederverwendbarkeit	Das Kontextmodell und die zugehörige Management-Komponente müssen so abstrakt und generisch gehalten werden, dass möglichst beliebige Kontextdaten bereitgestellt und abgefragt werden können.

Fortsetzung von Tabelle 5.7 auf der nächsten Seite ...

Kürzel	Titel	Details
KM3	Trennung von Modell und Management	Damit das Kontextmodell unabhängig von der konkreten Implementierung und somit allgemein einsetzbar ist, muss es kozeptionell unabhängig von der Management-Komponente sein. Nur so können Kontextdaten über Plattform- und Programmiersprachgrenzen hinweg genutzt werden.
KM4	Integrationsfähigkeit in bestehende Systeme	Das Kontextmodell sollte so abstrakt definiert sein, dass es in möglichst beliebige Anwendungen und Systemplattformen integriert werden kann.
KM5	Föderation verteilter Kontextdaten	Da nicht nur lokale Kontextdaten für Mobile Prozesse von Bedeutung sind, sondern auch die anderer Geräte in der Umgebung, muss ein Kontextmodell eine einheitliche Sicht auch auf entfernte Kontextinformationen bieten.
KM6	Abstraktion von konkreten Informationstypen	Zum Austausch von Kontextdaten und deren Nutzung in unterschiedlichen Anwendungen müssen die Kontextinformationen unabhängig von ihrer lokalen Repräsentation beschrieben und mit zusätzlichen Metainformationen ergänzt werden.
KM7	Möglichkeit zur Transformation von Kontextdaten	Die Management-Komponente muss, um flexibel einsetzbar zu sein und Redundanzen zu verhindern, Kontextdaten durch Transformation in unterschiedliche Repräsentationen überführen können.
KM8	Proaktive Kommunikation	Änderungen von lokalen oder entfernten Kontextdaten sollten proaktiv an interessierte Teilnehmer übermittelt werden, um flexibel und schnell auf diese Änderungen reagieren zu können.

Fortsetzung von Tabelle 5.7 auf der nächsten Seite ...

Kürzel	Titel	Details
KM9	Robustheit und Kompaktheit	Da mobile Geräte eher leistungsschwach sind und unter unzuverlässigen Rahmenbedingungen arbeiten, muss eine Kontextunterstützung möglichst leichtgewichtig und robust gegenüber Fehlern sein.
KM10	Flexibilität und Skalierbarkeit	Um das Kontextsystem möglichst optimal an Geräte und Situationen anpassen zu können, müssen funktional entkoppelte Komponenten, die flexibel kombiniert und erweitert werden können, vorgesehen werden.
KM11	Filter zum Abbilden von Kontextausschnitten	Da nur die Kontextdaten relevant sind, die von laufenden Anwendungen beziehungsweise Prozessen benötigt werden, kann durch den Einsatz von adäquaten Filtermechanismen die Menge der verwalteten Kontextdaten auf ein minimales Maß reduziert und so Ressourcen geschont werden.
KM12	Abbilden verschiedener Kontextabstraktionen	Da Kontextdaten neben Sensordaten auch abgeleitete Informationen, wie zum Beispiel Situationen, beschreiben, müssen unterschiedliche Abstraktionsgrade auch vom Kontextkonzept berücksichtigt werden.
KM13	Sicherheit und Vertrauen	Kontextdaten können sehr persönliche Informationen enthalten, die durch geeignete Sicherheits- und Vertrauensmaßnahmen geschützt werden müssen.
KM14	Einbeziehen der Qualität von Kontextdaten	Kontextdaten können je nach Quelle ungenau, unvollständig und sogar widersprüchlich sein. Um dies bei Entscheidungen berücksichtigen zu können, sind Angaben zur Qualität der verwendeten Kontextinformation wichtig.

Tabelle 5.7.: Anforderungen an die Kontextunterstützung (nach (KZTL08, Tur06))

5.6.2. Konzeption eines abstrakten und generischen Kontextmodells

Die kontextbasierte Kooperation mit ihren Mobilen Prozessen, die zwischen unterschiedlichen und heterogenen Geräten migrieren, erfordert damit also ein allgemeines und generisches Kontextkonzept, das unterschiedlichen Ansprüchen und Anforderungen gerecht werden muss (vgl. Abschnitt 5.6.1). Inwieweit bestehende Kontext-Plattformen diese Anforderungen erfüllen und wie ein spezielles Kontextmodell für die Ausführung Mobiler Prozesse aussehen kann, wird im Folgenden untersucht.

5.6.2.1. Defizite bestehender Kontext-Plattformen

Kontextmodelle (vgl. Abschnitt 3.3.2) und Middleware-Plattformen, welche diese Modelle umsetzen (vgl. Abschnitt 3.3.3), sind jeweils unterschiedlich gut geeignet, um die Anforderungen, die durch die kontextbasierte Kooperation an Kontextmodelle gestellt werden, zu erfüllen.

Konzepte, die auf einfachen Metamodellen – wie dem des *Key-Value-Modells* – basieren, ermöglichen es zwar, Kontextdaten besonders kompakt und somit leichtgewichtig abzubilden, jedoch ist es kaum möglich, zusätzliche Metainformationen zum Kontextdatum – wie semantische Verknüpfungen oder Qualitätsparameter – einzubinden. Deshalb ist es auch nicht möglich, unterschiedliche Kontextabstraktionen zu unterscheiden oder Sicherheits- und Vertrauensaspekte an einzelne Informationen zu binden. Auch die Skalierbarkeit und Flexibilität ist durch die fehlende Strukturierung stark eingeschränkt, da durch den relativ flachen Namensraum für die Werte des Key-Parameters schnell Namenskonflikte auftreten können. Insgesamt kann also gesagt werden, dass einfache Modelle wenig geeignet sind, um als Grund- beziehungsweise Metamodell für ein Kontextmodell zur Unterstützung Mobiler Prozesse zu dienen. Aus diesem Grund werden Plattformen, die solche einfachen Modelle zur Abbildung von Kontextdaten nutzen, im Folgenden nicht weiter betrachtet.

Ein Kontextmodell für kooperativ ausgeführte Mobile Prozesse muss also auf einem Metabeschreibungsmodell aufbauen, mit dessen Hilfe komplexere Kontextstrukturen und Metainformationen beschrieben werden können. Dabei muss jedoch auf ein ausgewogenes Verhältnis zwischen Ausdrucksmächtigkeit, Komplexität und Plattformunabhängigkeit geachtet werden. Hierbei sind auf *Ontologien* basierende und *logikbasierte Ansätze* zwar sehr ausdrucksstark, aber auch sehr ressourcenintensiv in ihrer Auswertung und Verarbeitung. Zusätzlich trennen gerade logikbasierte Ansätze nicht strikt zwischen Kontextmodell und -verwaltung, da

die Inferenzregeln, die zum Ableiten höherwertiger Kontextinformationen benötigt werden, bereits im Modell hinterlegt werden. Deshalb scheiden Ansätze wie die der *Gaia*-Middleware (vgl. Abschnitt 3.3.3.3), der *CONtext ONtology* (CONON) [WGZP04] oder der *Context Broker Architecture* (CoBrA) [CFJ03] ebenfalls zur Modellierung des Kontextmodells für Mobile Prozesse aus. [KZTL08, Tur06]

Gleichermaßen ungeeignet sind Ansätze, die sich auf einzelne Programmiersprachen und deren spezielle Kommunikationskonzepte beziehen, wie es beispielsweise vom *Java Context-Awareness Framework* (JCAF) [Bar05] getan wird, das serialisierbare *Java-Objekte* zur Modellierung des Kontexts und *Java Remote Method Invocation* (RMI) als Kommunikationsprotokoll verwendet. Ebenfalls nur eingeschränkt geeignet ist das *Hydrogen*-Rahmenwerk [HSP+03], das zwar sehr kompakt und robust ist sowie Kontextdateninformationen einheitlich bereitstellt, aber zur Kommunikation zwischen Anwendungen, Kontextquellen und Middleware auf TCP/IP beschränkt ist und zudem keine Möglichkeit der Transformation von Kontextinformationen bietet. Auch das *Context Toolkit* (vgl. Abschnitt 3.3.3.1), das sich durch die Verwendung von wiederverwendbaren *Kontext-Widgets* auszeichnet, bietet zwar einheitliche und erweiterbare Kontextinformationen, jedoch ist auch hier das Management und das Kontextmodell eng miteinander verwoben, sodass wenig Flexibilität gegeben ist. Zudem ist eine verteilte Föderation von Kontextdaten kaum möglich. [KZTL08, Tur06]

Da Mobile Prozesse in mobilen Peer-to-Peer- und Ad-hoc-Umgebungen Vorteile gegenüber zentral koordinierten Prozessen bieten und diese deshalb als Standardumgebung für die Konzepte zur Realisierung Mobiler Prozesse angesehen werden sollten, sind Kontextmodelle und -plattformen, die einen Großteil ihrer Kontextgewinnung und -verwaltung in unterstützende Infrastrukturkomponenten auslagern, ebenfalls nicht als Modellbasis geeignet. Hierzu zählen zum Beispiel die *Solar*-Plattform [CK02] oder das zusätzlich noch stark auf ortsbezogene Anwendungen ausgerichtete Kontextmodell der *Nexus*-Plattform (vgl. Abschnitt 3.3.3.2).

Zusammengefasst kann gesagt werden, dass keiner der betrachteten Ansätze zur Modellierung und Verwaltung von Kontextdaten für die Unterstützung der kooperativen Ausführung von Mobilen Prozessen in hoch dynamischen und mobilen Umgebungen uneingeschränkt geeignet ist. Vielmehr sind oftmals Kontextmodell und -verwaltung nicht klar getrennt, sodass ein flexibler Austausch der Kontextinformationen über heterogene Systeme hinweg sehr erschwert wird. Zudem werden oftmals nur spezielle Anwendungsdomänen, wie zum Beispiel ortsbezogene Anwendungen,

beziehungsweise einzelne Technologien und Paradigmen, wie Java RMI oder die Objektorientierung, unterstützt. Es fehlt jedoch ein Kontextmodell, das zwar die Kontextinformationen klar strukturiert und zum Aufbau föderierter Kontexte über unterschiedliche sowie heterogene Systeme und Plattformen genutzt werden kann, dabei jedoch die Gewinnung der Kontextinformationen und deren Verwaltung jeweils den lokalen Geräten überlässt. [KZTL08, Tur06]

Da ein Kontextmodell für Mobile Prozesse zudem a priori unbekannte Kontextdaten benötigt, muss ein entsprechendes Modell eben besonders allgemein gültig und unabhängig sein sowie einen Zugriff auf die enthaltenen Daten über möglichst viele Primärkontextinformationen erlauben. Aus diesen Gründen wird ein konzeptionelles Kontextmodell benötigt, das die genannten Eigenschaften und Voraussetzungen erfüllt.

5.6.2.2. Konzeptionelles Kontextmodell für Mobile Prozesse

Gerade in mobilen Umgebungen sind Ressourcen und Dienste, die Kontextinformationen liefern, meist sehr heterogen sowie unterscheiden sich in ihrer Repräsentation und Qualität. Zudem ist es kaum möglich, die von Mobilen Prozessen benötigten Kontextinformationen vorab festzulegen. Deshalb ist ein generisches Kontextmodell notwendig, mit dem Kontextinformationen aus unterschiedlichen Datenquellen in standardisierter Art und Weise zugreifbar gemacht werden. Ein konzeptionelles Kontextmodell, das dieses leisten kann, ist in Abbildung 5.8 dargestellt und wird im Folgenden vorgestellt:

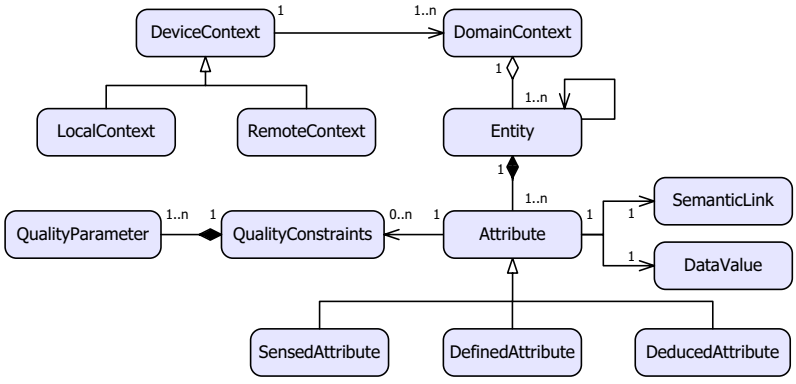


Abbildung 5.8.: Schematische Darstellung des generischen Kontextmodells (nach (KZTL08, Tur06))

Im Mittelpunkt des konzeptionellen Kontextmodells stehen einzelne Entitäten (*Entities*), die durch eine Menge von einzelnen Kontextinformatio-

nen und anderen Entitäten definiert sind. Sie beschreiben also höherwertige Kontextdaten als Aggregation einfacherer Informationen und geben so Kontextinformationen eine elementare Struktur, über die auf die Daten zugegriffen werden kann. Solch eine Entität kann zum Beispiel eine Person sein, die sich durch eine Identität, einen Aufenthaltsort und eine Aktivität auszeichnet oder ein Gebäude, das dann wiederum Räume mit ihren individuellen Eigenschaften als rekursive Entitäten enthält. [KZTL08, Tur06]

Innerhalb der Entitäten werden einzelne Kontextinformationen als *Attribute* abgebildet und stellen ein Merkmal der Entität dar. Die Summe aller Kontextattribute – auch rekursiv über Subentitäten bezogen – stellt somit den Kontext der Entität dar. Da diese Attribute aus unterschiedlichen und heterogenen Datenquellen stammen können, existieren auch unterschiedliche Attributtypen, die die Herkunft der Information näher erläutern. Kontextdaten, die direkt durch physikalische oder logische Sensoren erfasst wurden, werden als *Sensed Attribute* abgebildet und weisen in aller Regel auf ein sehr dynamisches Datum hin, das häufigen Änderungen unterliegt. Statische Attribute, die keinen oder nur seltenen Änderungen unterliegen, werden hingegen als *Defined Attributes* modelliert. Zu diesen Informationen zählt zum Beispiel die Identität eines Nutzers oder das Modell eines (mobilen) Gerätes. Attribute, die jedoch höherwertige Kontextinformationen darstellen, die aus anderen einfacheren Kontextdaten abgeleitet wurden, werden durch den dritten Attributtyp, dem *Deduced Attribute*, repräsentiert. [KZTL08, Tur06]

Allen Kontextattributen – unabhängig von ihrem Typ – ist gemein, dass sie einen Wert repräsentieren, der über die *Data-Value*-Komponente des Attributs zugegriffen werden kann und einen beliebigen einfachen oder komplexen Datentyp darstellt. Zudem wird das Attribut durch die Angabe eines *semantischen Links* – zum Beispiel in Form einer URI – mit Metadaten angereichert, die es dem Nutzer beziehungsweise einer Anwendung oder Verwaltungskomponente erlauben, die Semantik des Datums besser zu erfassen und gegebenenfalls Transformationen in äquivalente Repräsentationsformen durchzuführen. [KZTL08, Tur06]

Da Kontextinformationen – je nach Herkunft und Aktualisierungszeitpunkt – unpräzise, inkonsistent oder unvollständig sein können, ist es sehr sinnvoll, Angaben zur Qualität des Kontextattributes zu machen, um die Zuverlässigkeit der Kontextinformation bestimmen zu können. Deshalb kann jedem Attribut ein individueller Satz an Qualitätsparameter (*Quality Parameters*) in Form einer *Quality-Constraints*-Komponente zugeordnet werden. Diese können dann zum Beispiel die Abweichung einer

Ortskoordinate oder deren letzten Aktualisierungszeitpunkt beschreiben. [KZTL08, Tur06]

Um die Kontextinformationen weiter zu strukturieren und einen Filtermechanismus zum Erstellen von für einzelne Anwendungen relevante Weltausschnitte vorzuhalten, enthält das konzeptionelle Kontextmodell ein Domänenkonzept. Diese Domänen (*Domain Context*) enthalten jeweils eine begrenzte Anzahl von Entitäten und fassen sie zu einer logischen Einheit zusammen. Durch die Verwendung von Domänen ist es möglich, Kontextinformationen wiederzuverwenden und zugleich individuelle Kontextausschnitte bereitzustellen. [KZTL08, Tur06]

Alle Domänen eines Geräts werden dann im lokalen Gerätekontext (*Local Context*) zusammengefasst und bilden damit den lokal administrierten Teil des Gesamtkontextes dieses Systems. Ergänzt wird dieser Teil des Kontextes durch die lokalen Kontexte anderer (erreichbarer) Geräte der Umgebung, indem diese als *Remote Context* mit in das Gesamtkontextmodell aufgenommen werden. Durch diese Vereinigung der Kontextdaten mehrerer Geräte kann aus dem System heraus auf unterschiedliche Kontextdaten zugegriffen werden, auch wenn das eigene Gerät gewisse Kontextinformationen selbst nicht bereitstellen kann. [KZTL08, Tur06]

5.6.3. Integration einer verteilten Registratur zur Auswahl und Nutzung von Dienstinstanzen

Für die Ausführung Mobiler Prozesse besonders wichtige Kontextinformationen sind jene, die über die in der Umgebung verfügbaren Dienstinstanzen Auskunft geben. Deshalb ist die Suche nach Diensten anhand der in der Prozessbeschreibung vorgegebenen Dienstklasse und den einschränkenden nicht-funktionalen Bedingungen auf der Basis von Kontextinformationen eine elementare Aufgabe, die vom Kontext einer Umgebung für die Ausführung Mobiler Prozesse geleistet werden muss (vgl. Abbildung 5.9). Dabei werden die möglichen *abstrakten technologieabhängigen Dienstbeschreibungen* durch die Fähigkeiten des lokalen Gerätes eingeschränkt. Das Bestimmen der *konkreten Dienstinstanzen* wird dann von dem Vermögen des (entfernten) Diensterbringers zum Einhalten der geforderten nicht-funktionalen Parameter determiniert. Die hierbei notwendige enge Kopplung von Kontextwissen und Dienstverzeichnis führt dazu, dass eine verteilte Registratur als integrale Komponente des Kontexts anzusehen ist.

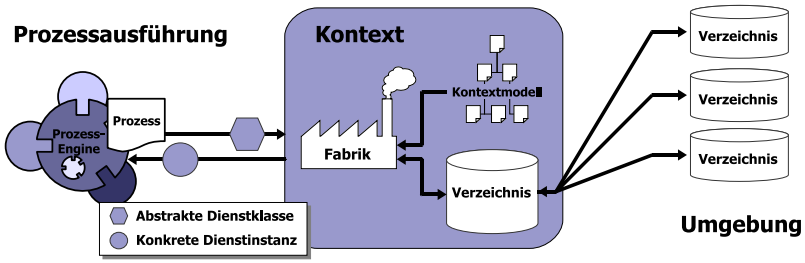


Abbildung 5.9.: Verzeichnisdienst als integraler Bestandteil des Kontexts

5.6.3.1. Anforderungen an einen Verzeichnisdienst im Umfeld Mobiler Prozesse

Die kontextbasierte Kooperation in Form von Mobilen Prozessen stellt besondere Anforderungen an einen Verzeichnisdienst (vgl. Tabelle 5.8), die erfüllt sein müssen, um möglichst optimal in das Gesamtkonzept integriert werden zu können. Dabei muss ein entsprechender Verzeichnisdienst als ein verteiltes System gleichberechtigter Partner ausgelegt sein (VV1). Dies begründet sich daraus, dass die kooperative Ausführung Mobiler Prozesse ohne zentrale Komponenten auskommt und so auch ein Verzeichnis zur Unterstützung der Ausführung ohne solche auskommen muss, damit der wesentliche Charakter dieser Kooperationsform nicht verloren geht.

Die hohe Dynamik mobiler verteilter Systeme führt dazu, dass es kaum möglich ist, einen stets aktuellen Datenbestand über angebotene Dienste anderer Geräte der Umgebung aufrecht zu erhalten. Deshalb sollten, auch wegen der begrenzten Ressourcen der Geräte, lediglich lokale Daten in den *Repositories* gespeichert werden und Informationen über entfernte Dienste nur bei Bedarf angefordert werden (VV2). Insgesamt sollten die Verwaltungsdaten, die zwischen den einzelnen Verzeichnissen einer mobilen Umgebung ausgetauscht werden müssen, minimal gehalten werden, damit die meist schmalbandigen und unzuverlässigen Kommunikationsverbindungen nicht unnötig beansprucht werden (VV3).

Ein Verzeichnis zur Verwaltung und Suche von Dienstinstanzen zur Ausführung von (Teil-)Aufgaben Mobiler Prozesse muss diese über *abstrakte Dienstklassen* (vgl. Abschnitt 5.5.4) referenzieren, da diese die Verbindung zwischen Aktivität und Dienstinstanz im Prozessmodell darstellen (VV4). Dabei führt dieser Zugriff über abstrakte Dienstklassen und die Konzentration auf lokale Daten zur Möglichkeit der Vereinfachung grundlegender Datenstrukturen des Verzeichnisses. [Ada06]. Die Verwendung von Dienstklassen bewirkt auch, dass bei der Auslieferung von *instanzspezifischen Dienstbeschreibungen* als Ergebnis einer Suche keine Annahmen über die

enthaltenen Daten oder deren Format gemacht werden dürfen (VV5). Dies ergibt sich aus der strikten Technologieneutralität der Dienstklassen, sodass prinzipiell jede Implementierungstechnologie vom Verzeichnis unterstützt werden muss.

Insgesamt muss also ein verteiltes Metaverzeichnis eingebunden werden, das die Eigenschaften Mobiler Systeme bezüglich Ressourcenarmut, unzuverlässiger Kommunikationsverbindungen sowie Heterogenität und Dynamik berücksichtigt. Dabei müssen neben dem Suchen über die Dienstklasse auch die nicht-funktionalen Bedingungen als Suchmerkmale berücksichtigt werden, sodass eine enge Kopplung an das Kontextkonzept gegeben ist (VV6). Hierbei bietet es sich an, auch die Suche selbst verteilt auszuführen, um lokale Ressourcen zu schonen und den Kommunikationsaufwand zu minimieren. [KZL07a]

Kürzel	Titel	Details
VV1	Verteiltes Verzeichnis	Da das Verzeichnis Teil eines kooperativen Systems gleichberechtigter Teilnehmer ist, muss auch dieses als verteilte Architektur ohne zentrale Steuerungselemente auskommen.
VV2	Lokalität der Daten	Aufgrund der hohen Dynamik in mobilen Umgebungen und der häufig sehr beschränkten Ressourcen mobiler Systeme ist es sinnvoll, lediglich lokale Daten im lokalen *Repository* des Verzeichnisses zu speichern. Daten über entfernte Dienste werden nur bei Bedarf angefordert.
VV3	Minimierung der Kommunikation	Durch die hohe Dynamik und die damit verbundenen hohen Änderungsraten sollten außerhalb von konkreten Anfragen nur wenig relativ statische Daten aktiv zwischen den Verzeichnissen einer mobilen Umgebung ausgetauscht werden.
VV4	Unterstützung abstrakter Dienstklassen	Da Mobile Prozesse Dienste über abstrakte Dienstklassen referenzieren, müssen diese im Verzeichnisses abbildbar sein.

Fortsetzung von Tabelle 5.8 auf der nächsten Seite ...

Kürzel	Titel	Details
VV5	Technologieneutrale Bereitstellung von Dienstbeschreibungen	Da das Verzeichnis unterschiedliche Implementierungstechnologien für Dienste unterstützen muss, muss die Auslieferung der Dienstbeschreibungen technologieneutral erfolgen, damit keine Technologie durch das Verzeichnis diskriminiert wird.
VV6	Einbinden des Kontexts	Durch die enge Kopplung von Kontextwissen und Verzeichnis durch die Verwendung nichtfunktionaler Bedingungen zur Auswahl von Dienstinstanzen müssen Kontext und Verzeichnis eng miteinander verbunden werden.

Tabelle 5.8.: Anforderungen an ein Verzeichnisdienst für Mobile Prozesse (z.T. nach (Ada06))

5.6.3.2. Defizite bestehender Systeme zur Dienstsuche

Verzeichnissysteme zur Dienstsuche und -bereitstellung stellen ein etabliertes Konzept innerhalb verteilter Systemarchitekturen dar. Dementsprechend vielfältig ist die Anzahl von unterschiedlichen Protokollen und Implementierungen, sodass an dieser Stelle lediglich eine beispielhafte Auswahl von exemplarischen Systemen anhand der zuvor aufgestellten Kriterien (vgl. Abschnitt 5.6.3.1) bewertet werden kann. Dabei werden der Web-Service-Standard *Universal Description, Discovery, and Integration* (UDDI, vgl. Abschnitt 4.1.2), die *Jini Network Technology* [Sun01], das P2P-basierte *Project JXTA* [Wil02] sowie das *Salutation-* [LH02] und das *Konark-System* [HDVL03] untersucht und evaluiert (vgl. Tabelle 5.9).

Klassische Verzeichnisse für verteilte Dienst- beziehungsweise Objektsysteme, wie zum Beispiel *UDDI* und *Jini*, besitzen meist eine zentral aufgebaute Architektur. Dabei besteht *UDDI* zwar prinzipiell aus mehreren Zugangsservern, die jedoch lediglich einen replizierten zentralen Datenbestand zugänglich machen. Bei *Jini* werden, aufgrund der Ausrichtung auf abgeschlossene Netzwerkumgebungen, für einzelne lokale Netzwerke jeweils isolierte und zentral aufgebaute Verzeichnisse, so genannte *Lookup-Services*, verwendet. Die für Peer-to-Peer-Systeme beziehungsweise dynamische Umgebungen konzipierten Protokolle und Architekturen zur Suche von Diensten wie *JXTA*, *Salutation* und *Konark* sind hingegen verteilt

aufgebaut und erfüllen so die Anforderung nach einer verteilten Verzeichnisstruktur (VV1). Zentral strukturierte Verzeichnissysteme subsumieren natürlich zwangsweise hauptsächlich Dienste anderer Geräte, sodass die Forderung nach der Konzentration auf die Bereitstellung von Daten über lokale Dienste (VV2) hier keine Anwendung finden kann. Bei den verteilten Verzeichnissystemen bieten *JXTA* und *Salutation* einen hohen Grad an Lokalität der bereitgestellten Daten an. Anfragen werden hierbei zwischen den bekannten Verzeichnissen einer Umgebung weitergereicht und so verteilt verarbeitet. Das *Konark*-System geht jedoch einen Schritt weiter und speichert lokal auch Dienste anderer, die entweder aufgrund einer entsprechenden Anfrage gefunden wurden oder die durch einen Server aktiv im System bekannt gemacht wurden, sodass hier eine strikte örtliche Trennung der Daten von lokalen und entfernten Diensten nicht mehr gegeben ist.

Kürzel	Titel	UDDI	Jini	JXTA	Salutation	Konark
VV1	Verteiltes Verzeichnis	−	○	+	+	+
VV2	Lokalität der Daten	⊗	⊗	+	+	−
VV3	Minimierung der Kommunikation	−	−	+	+	○
VV4	Unterstützung abstrakter Dienstklassen	○	−	−	−	−
VV5	Technologieneutrale Bereitstellung von Dienstbeschreibungen	−	−	⊗	−	−
VV6	Einbinden des Kontexts	○	○	−	○	−

\+ = unterstützt − = nicht unterstützt ○ = teilweise unterstützt ⊗ = entfällt

Tabelle 5.9.: Bewertung bestehender Verzeichniskomponenten

Der Kommunikationsaufwand bei den zentral aufgebauten Verzeichnissen und bei *Konark* ist im Verhältnis zu *JXTA* und *Salutation* höher, da hier zusätzlich zu Suchanfragen auch Dienstbeschreibungen zur Registrierung der Dienste – zumindest teilweise – über das Netzwerk versendet werden. Bei den Konzepten mit einer reinen lokalen Datenhaltung entfällt hingegen die Kommunikation zur Registrierung von Diensten über das

Kommunikationsnetzwerk und die Anforderung nach einer möglichst minimalen Kommunikation zur Bereitstellung des Verzeichnisdienstes (VV3) ist erfüllt.

Da das Konzept der Nutzung abstrakter Dienstklassen zur Beschreibung und Suche von Aktivitäten eines (Mobilen) Prozesses (VV4) bisher noch nicht verwendet wurde, unterstützen die betrachteten Systeme die Suche über Dienstklassen nicht direkt. Einzig *UDDI* bietet, da hier prinzipiell beliebige Daten zur Beschreibung und Suche eines Dienstes verwendet werden können, eine eingeschränkte Möglichkeit zur Verwendung von abstrakten Dienstklassen zur Dienstauswahl. Noch schlechter sieht die Unterstützung einer technologieneutralen Bereitstellung von Diensten (VV5) aus, die von keinem System bereitgestellt wird. Vielmehr sind die betrachteten Verzeichnisse – bis auf *JXTA*, das die Bereitstellung von Dienstbeschreibungen überhaupt nicht unterstützt, sondern nur das Vorhandensein eines Dienstes anzeigen kann – auf eine Technologie zur Dienstnutzung ausgerichtet. So verwendet zum Beispiel *UDDI* Dienstbeschreibungen in Form von WSDL-Dokumenten beziehungsweise das *Konark*-System eine eigene Dienstbeschreibungssprache.

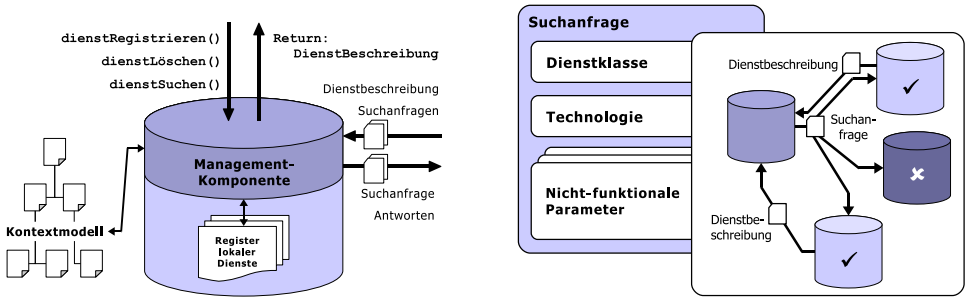
Auch die Einbindung von Kontextwissen zur Auswahl von Diensten wird von keinem der Systeme direkt unterstützt. Es ist jedoch bei einzelnen Systemen möglich, Kontextparameter auf vorhandene Strukturen abzubilden und so zu integrieren. So können bei *UDDI*, wiederum aufgrund der Möglichkeit zur Integration allgemeiner Beschreibungsattribute, Kontextdaten integriert werden, doch durch die statisch ausgerichtete Struktur des replizierten Datenbestandes kann eine dynamische Änderung der verwendeten Daten kaum stattfinden. Bei *Jini* werden zur Dienstsuche so genannte *Templates* verwendet, mit deren Hilfe auch Kontextdaten als Suchkriterium abgebildet werden können, jedoch ist auch hier ein dynamisches Update der Daten eher problematisch. Ähnlich sieht es bei *Salutation* aus, wobei hier jedoch zur Beschreibung eines Dienstes so genannte *FunctionalUnits* mit beschreibenden Attributdatensätzen verwendet werden, die die Kontextdaten aufnehmen könnten. Hier steht jedoch der dynamischen Änderung von Kontextdaten vor allem die vorgesehene Replikation der Datensätze auf unterschiedlichen Geräten entgegen.

Insgesamt kann gesagt werden, dass keines der betrachteten Systeme als Verzeichnisdienst für Mobile Prozesse in Betracht gezogen werden kann, ohne große Änderungen an den Konzepten vornehmen zu müssen. Deshalb bietet es sich an, ein erweitertes und auf die Nutzung von abstrakten Dienstbeschreibungen ausgerichtetes Konzept zu entwerfen. Da-

bei können jedoch Konzepte zur Verteilung des Verzeichnisses und der verteilten Anfrageverarbeitung, wie sie zum Beispiel von *JXTA* oder *Salutation* genutzt werden, aufgegriffen werden.

5.6.3.3. Konzept eines Verzeichnisses für Mobile Prozesse

Aus der Analyse bestehender Verzeichnissysteme (vgl. Abschnitt 5.6.3.2), ergibt sich, dass keines der untersuchten Systeme geeignet ist, in toto als Verzeichnisdienst zur Unterstützung der Ausführung Mobiler Prozesse zu dienen. Deshalb wird hier auf Basis der gewonnenen Erkenntnisse ein konsolidiertes Konzept für ein Metadienstverzeichnis zur Verwaltung von abstrakten Dienstklassen und zugehörigen Dienstinstanzen entwickelt (vgl. Abbildung 5.10).



(a) Verzeichniskomponenten (b) Verteilte Dienstsuche

Abbildung 5.10.: Verteilter Verzeichnisdienst für Mobile Prozesse

Das grundlegende Konzept des Verzeichnisses greift den vom Kontextmodell bekannten Grundgedanken der Föderation lokaler Datenbestände einzelner Geräte zu einem Gesamtverzeichnis wieder auf (vgl. Abschnitt 5.6). Ähnlich zu *JXTA* und *Salutation* besitzt jedes Gerät ein lokales Verzeichnis, in dem jedoch alle Daten über die abstrakten Dienstklassen und die technologieabhängigen Beschreibungen der vom lokalen System angebotenen Dienste aufgenommen werden. Bei Bedarf, also wenn keine passenden lokalen Dienste zur Verfügung stehen, werden, sofern vorhanden, die entsprechenden Informationen von Diensten erreichbarer Systeme der Umgebung temporär integriert.

Aufbau des lokalen Verzeichnisses Um das lokale Verzeichnis möglichst kompakt halten zu können, ist es der Kontextkomponente zugeordnet und nutzt, um Redundanzen zu vermeiden, Teile des Kontextwissens, um seine Dienstleistung zu erbringen (vgl. Abbildung 5.10(a)). Dabei besteht das

Verzeichnis selbst aus zwei funktionalen Komponenten. Eine Komponente ist hierbei für die *Verwaltung des Datenbestandes* sowie für das Stellen und Beantworten von Anfragen zur *Suche entfernter Dienste* zuständig und die andere ist ein *Register zur Aufnahme der lokalen Verzeichnisdaten*, die nicht direkt dem Kontextmodell zuzuordnen sind. Das Register kann in diesem Zusammenhang, durch die Verwendung der abstrakten Dienstklassen als Zugriffsschlüssel und die Auslagerung nicht-funktionaler Aspekte in das Kontextmodell, einfach aufgebaut sein und ist prinzipiell entsprechend des schematischen Aufbaus der Dienstklassenhierarchie strukturiert (vgl. Abschnitt 5.5.4).

Föderative Dienstsuche Um der hohen Dynamik in mobilen Umgebungen und der begrenzten Ressourcen mobiler Systeme gerecht zu werden, werden Informationen über entfernte Dienste nicht aktiv zwischen den einzelnen Geräten repliziert, sondern nur bei Bedarf angefordert (vgl. Abbildung 5.10(b)). Hierbei werden Dienstanfragen durch ihre Dienstklasse, die vom anfragenden Gerät unterstützte Aufruftechnologie – wie zum Beispiel *Web Services*, *Corba* oder *Java RMI* – und eine Liste der zu erbringenden nicht-funktionalen Parameter beschrieben. Diese Suchanfrage wird an alle bekannten Verzeichnisse in der Umgebung versendet und dort lokal ausgewertet. Hierzu muss lediglich die Netzwerkadresse als einziges Datum über die entfernten Geräte bekannt sein. Führt der lokale Abgleich der angefragten Dienstklasse, Technologie und der nicht-funktionalen Parameter zu einem positiven Ausgang, so wird an den Anfragenden eine Antwort mit den dienstspezifischen Details gesendet. Bei einem negativen Ausgang der Auswertung hingegen wird keine Antwortnachricht gesendet. Aus den Antworten braucht sich der Anfragende dann lediglich eine beliebige auszuwählen, da jede Antwort unter den gestellten Rahmenbedingungen gleichwertige Dienstinstanzen repräsentiert, weil die geforderte Dienstklasse und alle nicht-funktionalen Rahmenbedingungen eingehalten werden. Deshalb können aufgrund dieser angenommenen Äquivalenz sehr einfache Auswahlverfahren, wie zum Beispiel *FCFS*[3], verwendet werden. Auch wenn dies nicht unbedingt zum optimalen Ergebnis führt, bleibt das Ergebnis doch in den geforderten Rahmenbedingungen. Um schließlich den Fall abzudecken, in dem kein passender Dienst gefunden wird, wird die Suche nach einem zuvor bestimmten *Timeout* ergebnislos beendet. Insgesamt wird so eine Parallelisierung der Bearbeitung der Suchanfrage erreicht, sodass insgesamt die Ressourcen des lokalen Gerätes geschont

[3] First-Come, First-Served

werden und Ergebnisse zur Suchanfrage schneller zur Verfügung stehen.

5.7. Transaktionen für migrationsbasierte Kooperationsformen

Um mit der Hilfe von Mobilen Prozessen möglichst auch (semantisch) atomare und konsistente Teilschritte von komplexen Aufgaben abbilden zu können, sind auch hier Transaktionskonzepte notwendig, die wie eine logische Klammer einzelne Aktivitäten des Prozesses zu einer Einheit verbinden. Durch diese Einheit wird dann die Verarbeitung dieser Aktivitäten untereinander so eng gekoppelt, dass die Ausführung der einzelnen Teilaufgaben von denen der anderen abhängt. Hierbei gilt jedoch – wie bereits in Abschnitt 4.3.5 gezeigt –, dass die klassischen (verteilten) Transaktionskonzepte für lang andauernde (Geschäfts-)Prozesse und somit auch für Mobile Prozesse eher ungeeignet sind. Deshalb ist es notwendig zu untersuchen, inwieweit die bestehenden erweiterten Konzepte auch auf Mobile Prozesse angewendet werden können beziehungsweise welche Konzepte als Basis für ein weiter entwickeltes Transaktionskonzept dienen können.

5.7.1. Anforderungen an Transaktionsmodelle für Mobile Prozesse

An ein Konzept zum Etablieren einer transaktional geschützten Ausführung von mehreren Aktivitäten eines Mobilen Prozesses werden unterschiedliche Anforderungen gestellt, die erfüllt sein sollten, damit der grundlegende Charakter der Mobilen Prozesse auch bei der Ausführung von Transaktionen erhalten bleibt (vgl. Tabelle 5.10 und [Hol07]). Hierbei gilt, dass Transaktionen für Mobile Prozesse eher den langlebigen Business-Transaktionen ähneln und deshalb nicht alle ACID-Eigenschaften erfüllen können (vgl. Abschnitt 4.3.5). Jedoch muss ein entsprechendes Transaktionskonzept auf jeden Fall gewährleisten, dass die *Atomarität* der Ausführung der Transaktion erreicht wird. Es müssen also alle an einer Transaktion beteiligten Aktivitäten entweder gemeinsam erfolgreich ausgeführt werden oder aber gemeinsam scheitern (TA1). Hinzu kommt, dass Transaktionen die *Konsistenz* des Prozesses gewährleisten müssen, damit der Mobile Prozess stets einen wohl definierten Zustand hat. Hierbei muss das Transaktionskonzept dafür Sorge tragen, dass bei der erfolgreichen genauso wie bei der gescheiterten Ausführung einer Transaktion immer von einem anfänglichen konsistenten Zustand in einen anderen ebenfalls konsistenten Zustand übergegangen wird (TA2).

Um die grundlegenden Merkmale Mobiler Prozesse auch unter der Verwendung von Transaktionen beizubehalten, muss ein Transaktionskon-

zept vor allem die *Migrationseigenschaft* der Mobilen Prozesse unterstützen. Das bedeutet also, dass während der Ausführung einer Transaktion der Prozess zwischen unterschiedlichen Geräten migrieren können muss, möglichst ohne dass hierdurch der Ablauf der Transaktion beeinflusst wird (TA3). Zudem muss das Konzept unter den schwierigen Randbedingungen in mobilen Umgebungen bezüglich Heterogenität und Dynamik der verfügbaren Teilnehmer in der Lage sein, *langlebige Transaktionen* auszuführen, die auch zeitlich ausgedehnte und gegebenenfalls manuelle Aktivitäten beinhalten (TA4). Zudem müssen unterschiedliche Dienstarten durch das Transaktionskonzept unterstützt werden, die neben ungeschützten Aktionen – ihnen fehlen die ACID-Eigenschaften außer der Konsistenz – und geschützten Aktionen – sie besitzen die ACID-Eigenschaften – auch Aktivitäten umfassen können, die physische Aktionen in der Realwelt repräsentieren (TA5). Insgesamt sollte das Transaktionskonzept hierbei keine zeitlich-logische Abfolge fordern, damit beliebige Aktivitäten des Mobilen Prozesses zu einer Transaktion zusammengefasst werden können und so *flexible Transaktionsbereiche* entstehen (TA6).

Kürzel	Titel	Details
TA1	Gewährleisten der Atomarität	Das Transaktionskonzept muss auch in Mobilen Prozessen gewährleisten, dass alle Aktivitäten einer Transaktion stets nur gemeinsam zum erfolgreichen Abschluss gebracht werden oder alle gemeinsam scheitern.
TA2	Sicherstellen der Konsistenz	Ein Transaktionskonzept für Mobile Prozesse muss dafür Sorge tragen, dass durch die Ausführung einer Transaktion stets ein konsistenter Zustand erreicht wird.
TA3	Unterstützung der Migrationseigenschaft	Da die kontextbasierte Kooperation durch die Migration der Mobilen Prozesse realisiert wird, muss ein Transaktionskonzept diese Eigenschaft unterstützen.

Fortsetzung von Tabelle 5.10 auf der nächsten Seite ...

Kürzel	Titel	Details
TA4	Unterstützung langlebiger Transaktionen	Durch die Integration lang andauernder (manueller) Aktivitäten in Mobile Prozesse muss ein Transaktionskonzept auch unter den schwierigen Randbedingungen mobiler Umgebungen eine Unterstützung von Transaktionen erlauben, die solche lang andauernden Aktivitäten beinhalten.
TA5	Unterstützung unterschiedlicher Dienstarten	Da Mobile Prozesse keine Einschränkungen bezüglich der Dienstarten machen, sollte auch ein Transaktionskonzept *ungeschützte*, *geschützte* und *reale Aktionen* unterstützen.
TA6	Unterstützung flexibler Transaktionsbereiche	Das Transaktionskonzept sollte möglichst keine Einschränkungen bezüglich der vorausgesetzten zeitlich-logischen Abfolge von Aktivitäten machen, um möglichst flexibel Transaktionen definieren zu können.
TA7	Toleranz gegenüber Knotenausfällen	Wegen der intrinsischen Eigenschaften mobiler Systeme und der potenziell langen Ausführungszeit von Mobilen Prozessen muss ein Transaktionskonzept mit dem Ausfall von einzelnen Ausführungsknoten umgehen können.
TA8	Minimierung der Kommunikation	Da in mobilen Systemen meist schmalbandige und unzuverlässige Kommunikationsanbindungen vorhanden sind, muss ein Transaktionskonzept für Mobile Prozesse einen möglichst geringen Kommunikationsaufwand haben.
TA9	Unabhängigkeit von Netzwerkstrukturen	Aufgrund der Unabhängigkeit Mobiler Prozesse von Netzwerktopologien und -technologien darf auch ein Transaktionskonzept möglichst keine speziellen Technologien oder Topologien fordern.

Tabelle 5.10.: Anforderungen an Transaktionskonzepte für Mobile Prozesse (z.T. nach (Hol07))

Aus den systemimmanenten Besonderheiten mobiler Umgebungen ergibt sich zudem, dass ein Transaktionskonzept für Mobile Prozesse möglichst eine hohe *Toleranz gegenüber Knotenausfällen* im System besitzen sollte (TA7). Dies bedeutet, dass ein Ausfall von Netzwerkknoten, die an der Ausführung des Mobilen Prozesses beteiligt sind oder waren, keinen Einfluss auf die Ausführung der Transaktion haben sollten. Schließlich sollte das Konzept so beschaffen sein, dass *wenige Kommunikationsakte* zur Koordinierung der Transaktion zwischen entfernten Teilnehmern ausgetauscht werden müssen, um die Ressourcen des lokalen Gerätes zu schonen und effizient mit den zur Verfügung stehenden Kommunikationsbandbreiten umzugehen (TA8). Hierbei ist es besonders wichtig, dass das Transaktionskonzept keine Voraussetzungen bezüglich zwingend benötigter Netzwerktopologien oder -technologien macht. Diese Notwendigkeit zur *Unabhängigkeit von Netzwerkstrukturen* (TA9) ergibt sich hierbei aus der strengen Technologieneutralität des Konzepts der Mobilen Prozesse, die damit dann auch transitiv für das Transaktionskonzept gelten muss.

5.7.2. Bewertung exemplarischer Transaktionsmodelle im Kontext Mobiler Systeme

Da die transaktionale Absicherung von strukturierten Vorgängen in (verteilten) Rechensystemen ein verbreitetes Konzept ist, existieren diverse Ausprägungen und Adaptionen des Konzepts der Transaktion für unterschiedliche Technologiebereiche und Systemumgebungen. Die große Vielzahl unterschiedlicher Modelle kann an dieser Stelle nicht vollständig betrachtet werden, sodass hier nur ein Vergleich grundlegender Modelle gegeben werden kann. Dabei wird auf die bereits in Abschnitt 4.3.5 eingeführten *klassischen ACID-Transaktionen* und *Kompensationssphären* sowie auf *atomare Sphären* [LR00] und *mobile Transaktionen* [HTKR05] eingegangen (vgl. Tabelle 5.11). Als mobile Transaktionen werden hierbei alle Transaktionskonzepte für infrastrukturbasierte mobile Systeme bezeichnet, die es mobilen Geräten erlauben, während einer Transaktion die Basisstationen zu wechseln (*Handover*).

Da die Gewährleistung der Atomarität (TA1) eines der primären Ziele aller betrachteten Transaktionskonzepte ist, wird diese Anforderung auch von allen untersuchten Konzepten erfüllt. Jedoch erfüllen nicht alle Ansätze die Forderung nach der Sicherstellung der Konsistenz (TA2).

Die betrachteten kompensationsbasierten Ansätzen der *Kompensationssphären* und der *mobilen Transaktionen* können nämlich die korrekte Ausführung der Kompensationshandlung nicht garantieren und da zusätzlich teilweise die Isolation bewusst aufgegeben wird, ist die Konsistenz bei diesen Konzepten nur eingeschränkt gegeben. [Hol07]

Die aus dem Konzept der Mobilen Prozesse abgeleitete Forderung der Unterstützung der Migrationseigenschaft (TA3) wird lediglich von den mobilen Transaktionen (zumindest bedingt) unterstützt. Alle anderen erfüllen aufgrund ihrer Ausrichtung auf statische und drahtgebundene Ausführungsumgebungen diese geforderte Eigenschaft nicht. Allerdings ist das Verständnis von Migration bei mobilen Transaktionen auf das Handover zwischen zwei Basisstationen begrenzt und stimmt somit nur teilweise mit dem Migrationsbegriff der Mobilen Prozesse überein, sodass eine vollständige Unterstützung dieser Anforderung nicht gegeben ist.

Langlebige Transaktionen (TA4) werden von den auf Sperrverfahren aufsetzenden klassischen ACID-Transaktionen und ihrer Erweiterung zu atomaren Sphären sehr schlecht unterstützt, da die belegten Ressourcen über die gesamte Laufzeit der Transaktion anderen nicht zur Verfügung stehen. Entsprechend anders sieht es dagegen bei den kompensationsbasierten Konzepten aus, die lang andauernde Transaktionen gut unterstützen. Bei der Forderung nach der Unterstützung unterschiedlicher Dienstarten (TA5) sieht es ähnlich aus, da auch hier die ACID-Transaktionen und die atomaren Sphären durch die Beschränkung auf die Verwendung von geschützten Diensten die geforderte Eigenschaft kaum unterstützen. Die mit Hilfe der Kompensation arbeitenden Transaktionsmodelle hingegen beziehen alle Dienstarten – ob ungeschützt, geschützt oder real – in ihr Konzept ein, sodass diese die Forderung wiederum voll erfüllen.

Da nur bei der Ausführung von Prozessen die Definition eines flexiblen Transaktionsbereiches (TA6) sinnvoll ist, muss sich hier die Betrachtung auf die für Workflows entworfenen Sphärenkonzepte beschränken. Dabei gilt, dass die Sphärenkonzepte prinzipiell diese Anforderung umsetzen. Jedoch erfüllen die atomaren Sphären aufgrund ihrer strukturellen Einschränkungen, die durch ihre Definition gegeben sind, die Forderung nach flexiblen Transaktionsbereichen nur eingeschränkt. [Hol07]

Bei der Forderung nach einer Toleranz gegenüber Knotenausfällen (TA7), ist wiederum die Zweiteilung in die durch Sperren eng gekoppelten klassischen ACID-Transaktionen und atomaren Sphären auf der einen sowie die kompensationsbasierten Sphären und mobilen Transaktionen

auf der anderen Seite gegeben. Die enge Kopplung zwischen den Knoten der ersten beiden Konzepte führt dabei dazu, dass die Toleranz gegenüber Knotenausfällen gering ist. Das Fallenlassen der engen Kopplung zu Gunsten der Kompensation führt bei den anderen beiden Konzepten dazu, dass diese hingegen Knotenausfälle tolerieren können. Ähnliches gilt bei der Forderung nach einer Minimierung der Kommunikation (TA8), da eng gekoppelte Konzepte meist einen sehr viel höheren Kommunikationsaufwand beinhalten, um die beteiligten Knoten mit ihren gegebenenfalls vorhandenen Sperren zu koordinieren. Deshalb gilt auch hier, dass ACID-Transaktionen und atomare Sphären hierbei schlecht abschneiden, wogegen Kompensationssphären und mobile Transaktionen diese Forderung erfüllen.

Eine vollständige Unabhängigkeit bezüglich geforderter Netzwerkstrukturen ist bei keinem der betrachteten Konzepte gegeben. Hierbei sind jedoch die mobilen Transaktionen durch ihre Ausrichtung auf infrastrukturbasierte Kommunikationsnetze am stärksten auf eine Netzwerkstruktur bezogen, sodass diese die genannte Eigenschaft am schlechtesten erfüllen. Die Sphärenkonzepte hingegen sind zwar unabhängiger, bauen aber auf einer zentrale Ausführungs-Engine auf, die alle Ausführungspfade des Prozesses kennt. Da aber bei Mobilen Prozessen autonome Prozesspfade entstehen können, erfüllen die Sphärenkonzepte die Anforderung nur bedingt. Analog entsprechen auch die klassischen verteilten ACID-Transaktionen mit ihren zentralen Koordinatoren zur Steuerung des Commit-Protokolls nur bedingt der Forderung nach Unabhängigkeit bezüglich geforderter Netzstrukturen. [Hol07]

Zusammenfassend lässt sich damit sagen, dass die Kompensationssphären – bis auf die nicht gegebene Unterstützung der Migration – gut geeignet sind, Mobile Prozesse um ein Transaktionskonzept zu erweitern. Deshalb bilden sie eine besonders vorteilhafte Grundlage zur Entwicklung eines entsprechenden erweiterten Transaktionskonzeptes im folgenden Abschnitt.

Kürzel	Titel	ACID	atomare Sphären	Kompensationssphären	mobile Transaktionen
TA1	Gewährleisten der Atomarität	+	+	+	+
TA2	Sicherstellen der Konsistenz	+	+	○	○
TA3	Unterstützung der Migrationseigenschaft	−	−	−	○
TA4	Unterstützung langlebiger Transaktionen	−	−	+	+
TA5	Unterstützung unterschiedlicher Dienstarten	−	−	+	+
TA6	Unterstützung flexibler Transaktionsbereiche	⊗	○	+	⊗
TA7	Toleranz gegenüber Knotenausfällen	−	−	+	+
TA8	Minimierung der Kommunikation	−	−	+	+
TA9	Unabhängigkeit von Netzwerkstrukturen	○	○	○	−

+ = unterstützt − = nicht unterstützt ○ = teilweise unterstützt ⊗ = entfällt

Tabelle 5.11.: Bewertung etablierter Transaktionskonzepte nach (Hol07)

5.7.3. Transaktionsunterstützung für Mobile Prozesse

Wie die Analyse bestehender Transaktionskonzepte (vgl. Abschnitt 5.7.2) gezeigt hat, sind klassische ACID-Transaktionen nicht als Basis für Transaktionen für Mobile Prozesse geeignet. Vielmehr sollte ein entsprechendes Konzept auf dem Modell der Kompensationssphären aufbauen, sodass als Voraussetzung für das hier vorgestellte Konzept gilt, dass die Aktivitäten innerhalb einer Transaktion durch entsprechende globale oder diskrete Gegenaktivitäten semantisch kompensierbar sind. Damit ist es dann möglich, in vielen Situationen auf einen über die gesamte Laufzeit erreichbaren Koordinator zu verzichten. Zudem sind neben einer solchen

Strategie zum Backward Recovery, wenn möglich, auch Forward-Recovery-Maßnahmen angebracht, um den Verlust bereits erbrachter Arbeit zu minimieren (vgl. Abschnitt 4.3.5).

Das für Mobile Prozesse entwickelte Transaktionskonzept ist dabei systematisch für unterschiedlich komplexe Prozessstrukturen entwickelt worden. Zunächst werden hierbei *Transaktionen über sequenzielle Prozesspfade* eingeführt, die das Grundmodell für weitere Transaktionskonzepte bilden. Darauf aufbauend wird dann auf ein Konzept zum Umgang mit *parallelen Prozesspfaden innerhalb einer Transaktion* eingegangen und abschließend werden noch *flexible Transaktionsbereiche* betrachtet, die eine beliebige Teilmenge der Aktivitäten eines Mobilen Prozesses zu einer Transaktion vereinen. [Hol07]

Sequenztransaktionen Sequenztransaktionen können über alle Prozessstrukturen definiert werden, die garantieren, dass beim Navigieren durch den Kontrollfluss ein sequenzieller Ausführungspfad entsteht. Dies können entweder im Kontrollfluss rein sequenziell angeordnete Aktivitäten sein oder aber Kontrollflussstrukturen umfassen, die zwar alternative Pfade aufweisen, diese aber durch entsprechende Kontrollflusskonstrukte – zum Beispiel *XOR-Splits* – in einem rein sequenziellen Ausführungspfad enden (vgl. Abbildung 5.11). Es darf also innerhalb einer Sequenztransaktion keine parallele beziehungsweise nebenläufige Ausführung von Aktivitäten geben. Die enthaltenen Aktivitäten dürfen dabei sowohl ungeschützt als auch geschützt sein, da sie sofort festschreiben dürfen. Es muss jedoch eine spätere Kompensation möglich sein. Durch das sofortige Festschreiben wird die Isolation der Transaktion zunächst aufgehoben, führt aber dazu, dass keine aufwendige Synchronisation notwendig ist und auch bezüglich der Migration des Mobilen Prozesses keine Einschränkungen entstehen. Dies hat zur Folge, dass jeweils das den Mobilen Prozess aktuell ausführende Gerät die gesamte Kontrolle über die erfolgreiche Ausführung beziehungsweise das Scheitern der Sequenztransaktion besitzt. [Hol07]

Zu Beginn einer Sequenztransaktion muss zunächst das gesamte Startset, also alle Informationen und Daten, die von den in die Transaktion eingehenden Kontrollflusskonstrukten benötigt werden, für einen eventuellen Neustart der Sequenztransaktion gesichert werden. Danach kann die Transaktion mit der Ausführung der enthaltenen Aktivitäten begonnen werden und im Falle einer gültigen Ausführungsfolge wieder erfolgreich beendet werden. Treten bei der Ausführung hingegen Fehlerfälle auf, sind

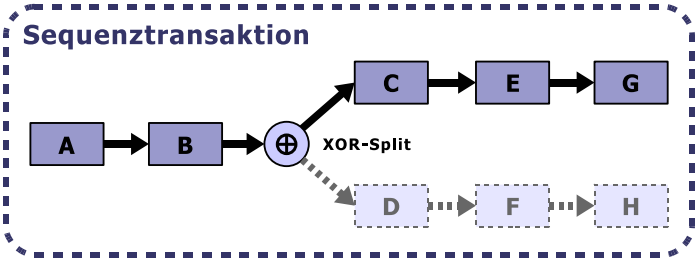


Abbildung 5.11.: Sequenzielle Transaktion (nach (Hol07))

von der Ausführungsumgebung entsprechende Recovery-Maßnahmen einzuleiten. Besteht der aufgetretene Fehler dabei in einer unzulänglichen Ausführung eines Dienstes, kann zunächst durch Forward Recovery versucht werden, die Aktivität durch ein erneutes Ausführen doch noch erfolgreich zu beenden und so die Transaktion fortzuführen. Scheitert die Ausführung einer Aktivität jedoch endgültig, da zum Beispiel keine Dienste oder Migrationspartner vorhanden oder zeitliche Restriktionen abgelaufen sind, so scheitert auch die Sequenztransaktion und muss kompensiert werden. Hierbei muss entweder die eine globale Kompensationsaktivität ausgeführt werden oder aber es muss bei diskreter Kompensation ein entsprechender Kompensationsprozess erzeugt werden. Hierbei kann es vorkommen, dass gezielt einzelne Dienstinstanzen adressiert werden müssen, da nicht in jedem Fall Dienst und Kompensationsdienst unabhängig voneinander sind. Zu diesem Zweck muss innerhalb der Transaktion an den entsprechenden Stellen die Adresse der Dienste, die entsprechende Aktivitäten ausgeführt haben, protokolliert werden. [Hol07]

Verlassen mehr als ein Prozesspfad die Sequenztransaktion, kann es sinnvoll sein, diese ausgehenden Pfade erst nach Abschluss der Transaktion zu betreten, um in Fehlerfällen keine Inkonsistenzen zu erhalten, die durch die Kompensation der Transaktion und ihr erneutes Ausfühen entstehen können. Die Notwendigkeit eines solchen *Backout-Schutzes*, der verhindert, dass der Kontrollfluss die Transaktion vor ihrem Abschluss verlässt, ist dabei anwendungsabhängig und muss deshalb vom Modellierer eines Mobilen Prozesses gegebenenfalls selbst aktiviert werden. [Hol07]

Join-Transaktionen Um nicht nur die Aktivitäten eines sequenziellen Prozesspfades durch Transaktionen zu koppeln, sondern auch parallele Strukturen zu unterstützen, bietet das Transaktionskonzept für Mobile Prozesse zusätzlich *Join-Transaktionen* an.

Eine Join-Transaktion macht sich die Eigenart von Mobilen Prozessen zu Nutze, dass das Zusammenführen paralleler Prozessstränge stets an einem eindeutig definierten Ort stattfindet. Somit ist es auch möglich, an diesem *Join-Knoten* die Ergebnisse parallel ausgeführter Sequenztransaktionen auszuwerten und zu einer gemeinsamen Transaktion zusammenzufassen (vgl. Abbildung 5.12). Selbst bei einem endgültigen Scheitern wird im Zuge der Synchronisation und Dead-Path-Eliminierung das Ergebnis an den Join-Knoten übermittelt, der auch die anderen parallelen Sequenztransaktionen zurück setzen kann. [Hol07]

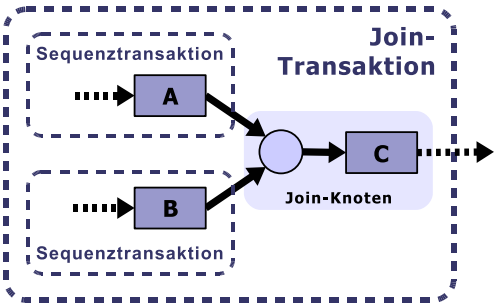


Abbildung 5.12.: Join-Transaktion (nach (Hol07))

Sequenz- und Join-Transaktionen können dabei nahezu beliebig geschachtelt werden und so auch komplexe transaktionale Strukturen abbilden. Durch das Einfügen einer so genannten Routing-Aktivität, die keinen Dienst referenziert, sondern lediglich eine künstliche Synchronisierung eigentlich nicht zusammenfließender paralleler Pfade bewirkt, können auch solche unverbundenen Prozessstränge in eine Transaktion aufgenommen werden. Durch Join-Knoten kann also eine externe Koordinierung der beteiligten Transaktionen vermieden werden. [Hol07]

Das endgültige Scheitern einer Join-Transaktion hängt dabei von den Join-Bedingungen ab, sodass nicht unbedingt das Scheitern einer Subtransaktion auch zu einem Abbruch der übergeordneten Transaktion führen muss. Scheitert eine Join-Transaktion jedoch, kann wiederum durch Forward Recovery versucht werden, einzelne Subtransaktionen erneut auszuführen oder aber das Backward Recovery führt zur Kompensation aller beteiligten Transaktionen, sodass sich die Abbruchentscheidung rekursiv über alle geschachtelten Transaktionen fortsetzt.

Flexible Transaktionsbereiche Flexible Transaktionsbereiche stellen eine weitere denkbare Variante von Transaktionen in Mobilen Systemen dar. Bei diesen handelt es sich um Sequenz- und Join-Transaktionen, die nicht

hierarchisch angeordnet sind, sondern beliebige Untermengen der Aktivitäten eines Mobilen Prozesses zu einer Transaktion zusammenfassen und somit ein Höchstmaß an Flexibilität bieten. Da solche Transaktionsbereiche jedoch nicht mehr an bestehende Kontrollflusskonstrukte der Mobilen Prozesse gebunden sind, können auch keine diesen Prozessen immanente Synchronisationsstrukturen zur Koordination der Transaktion genutzt werden. [Hol07]

Aus diesem Grund können flexible Transaktionsbereiche nicht effizient innerhalb Mobiler Prozesse genutzt werden, ohne wesentliche Eigenschaften wie zum Beispiel die Autonomie der beteiligten Geräte fallen lassen zu müssen. Denn nur durch das Einführen eines Transaktionskoordinators können solche separierten Transaktionsbereiche synchronisiert und verwaltet werden, sodass sie möglichst nicht in Mobilen Prozessen verwendet werden sollten.

Insgesamt kann also festgestellt werden, dass durch die Sequenz- und die Join-Transaktion ein hohes Maß an transaktionalem Verhalten auch in Mobile Prozesse integriert werden kann. Allerdings kann eine maximale Flexibilität bei der Definition von Transaktionsgrenzen nicht ohne Abstriche an den Kerncharakterzügen der Mobilen Prozesse erreicht werden. Im Allgemeinen reichen Join- und Sequenztransaktionen jedoch aus, um gängige Transaktionsmuster auch in mobilen Kontexten ausreichend zu realisieren.

6. Architektur und Implementierung einer Systemplattform für Mobile Prozesse

Der Ansatz der kontextbasierten Kooperation und dessen konzeptionelle Umsetzung als Mobile Prozesse ermöglichen es den Teilnehmern einer mobilen Umgebung, deren Potenziale durch Kooperation in einem größeren Umfang zu nutzen als es ohne eine solche Kooperation möglich ist (vgl. Kapitel 5). Mit Hilfe von Mobilen Prozessen ist es so selbst Geräten, die kaum eigene Dienste bereitstellen, möglich, komplexe Aufgaben eines Benutzers anzustoßen und damit als Zugangspunkt zu den Ressourcen und Diensten des mobilen Gesamtsystems zu dienen. Um die eingeführten Methoden und Ansätze für unterschiedliche mobile Geräte auch praktisch nutzbar zu machen, wird ein Umsetzungskonzept benötigt, das möglichst flexibel an die Bedürfnisse und Fähigkeiten einzelner mobiler Systeme angepasst werden kann. Dementsprechend führt dieses Kapitel ein Konzept für eine komponentenbasierte Systemplattform ein, welche die kontextbasierte Kooperation in Form von Mobilen Prozessen realisiert.

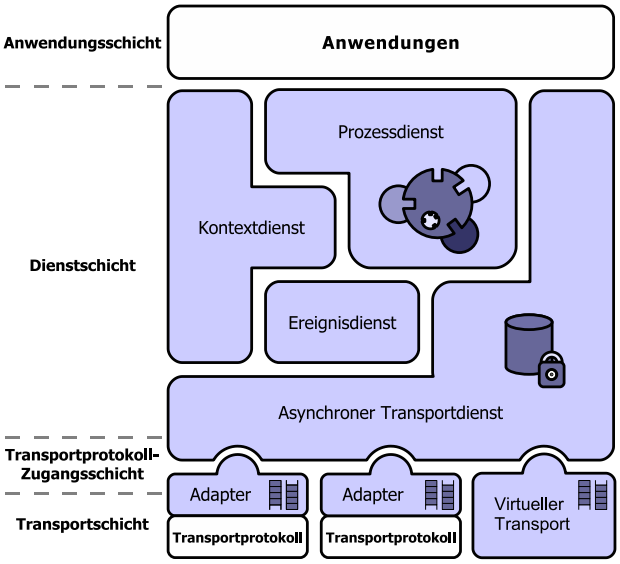


Abbildung 6.1.: Funktionale Komponenten der Grobarchitektur (nach (KZL07b, KZL06))

Die Architektur dieser Systemplattform (vgl. Abbildung 6.1) ist als Middleware konzipiert und ordnet sich damit als Dienstschicht zwischen An-

wendungs- und Transportschicht der beteiligten Systeme ein[1]. Grundsätzlich muss dabei eine Systemplattform für mobile Systeme so konzipiert sein, dass sie sich flexibel an die Bedürfnisse und Fähigkeiten des mobilen Geräts anpassen kann. Deshalb muss eine entsprechende Architektur durchgehend in lose gekoppelte funktionale Komponenten gegliedert sein, sodass durch eine geeignete Auswahl von Teilsystemen eine möglichst optimal abgestimmte Plattform entsteht, welche die Leistungsmerkmale des Gerätes und die Anforderungen der auf ihm laufenden Anwendungen abdeckt. Dabei muss es möglich sein, nicht benötigte oder nicht unterstützte Komponenten wegzulassen, ohne dass die restlichen Systemteile davon in ihrer Funktion maßgeblich beeinträchtigt werden. Eine solche lose Kopplung von autonomen Teilsystemen bietet dann sogar die Möglichkeit, die Systemplattform während der Laufzeit dynamisch an veränderte Umgebungsbedingungen anzupassen und so Kontextbewusstsein (vgl. Abschnitt 2.5.2 und 3.3) als ein Hauptmerkmal moderner Systeme im Mobile Computing auch auf der Ebene der Middleware zu etablieren. Denn nur durch Mechanismen zur dynamischen Adaption der Plattform ist es möglich, die in mobilen Systemen bestehende große Dynamik und Heterogenität zu überwinden.

Diesem Ansatz folgend kann eine Middleware-Plattform zur Ausführung Mobiler Prozesse in vier funktionale Komponenten, die jeweils eine gekapselte Dienstleistung für das Gesamtsystem erbringen, unterteilt werden. Dabei muss zunächst eine einheitliche Kommunikationsplattform etabliert werden, die unterschiedliche Transportmechanismen und -protokolle kapselt und so eine homogene Grundlage für weitere Komponenten bietet und die bestehende Heterogenität soweit notwendig verbirgt. Diese Aufgabe wird in der vorgestellten Architektur durch den *asynchronen Transportdienst* erfüllt. Des Weiteren muss eine Systemplattform für mobile Systeme in der Lage sein, Informationen über Zustandsänderungen einfach und zuverlässig anderen Systemen der mobilen Umgebung zur Verfügung zu stellen. Deshalb ist die Integration von proaktiven Kommunikationsformen in eine Middleware zur Ausführung Mobiler Prozesse besonders wichtig. Dementsprechend ist ein *Ereignisdienst*, der ereignisbasiert Informationen im Gesamtsystem der mobilen Umgebung verteilen kann, ein wichtiger weiterer Bestandteil der vorgestellten Middleware. Da die Ausführung von Mobilen Prozessen bei der Wahl von Migrationspartnern und bei der Ausführung von Aktivitäten stark von Kontextinformationen

[1] Die Bezeichnung der Schichten ist funktional gewählt und nicht analog zum ISO/OSI-Referenzmodell zu verstehen.

abhängt, ist eine entsprechende Komponente zur Verwaltung von Umgebungsinformationen unablässig für eine Middleware zur Realisierung der kontextbasierten Kooperation. Deshalb müssen die beiden Basisdienste zur Kommunikation um einen entsprechenden *Kontextdienst* zur Abbildung und zum Management von Kontextdaten ergänzt werden. Zudem muss ein *Prozessdienst* in die Architektur integriert sein, der die abstrakten Beschreibungen der Mobilen Prozesse interpretieren und ausführen kann. Eine entsprechende prototypische Implementierung dieser Komponenten wurde dabei im Projekt *Distributed Environment for Mobility-Aware Computing*[2] (DEMAC) realisiert und als Basis für eine Evaluierung des Ansatzes herangezogen. Diese Umsetzung ist für die *Java-Plattform, Micro Edition*, mit *Connected Device Configuration* und *Personal Profile* geschrieben und läuft auf der virtuellen Maschine J9[3]. Da diese Java-Version jedoch eine Untermenge des Sprachumfangs der Standardsprache ist, kann die Implementierung für leistungsstarke Rechner auch auf der *Java-Plattform, Standard Edition*, ausgeführt werden.

In den folgenden Abschnitten des Kapitels wird auf die oben genannten Dienste der Systemplattform und deren internen Aufbau näher eingegangen. Hierbei sind diese jeweils entsprechend dem *Singleton*-Erzeugungsmuster [GHJV03] konzipiert, sodass sichergestellt ist, dass jeder Dienst nur als ein Exemplar pro System existiert und somit ressourcenintensive Redundanzen vermieden werden. Bei der Darstellung der Komponenten werden jeweils grundlegende Aspekte der Architektur des Diensts und seiner Komponenten herausgearbeitet und ein Einblick in die prototypische Implementierung der Teilsysteme gegeben.

6.1. Das Kommunikationssystem

Das Kommunikationssystem der Systemplattform greift bekannte Paradigmen zur asynchronen Kommunikation in mobilen Systemen auf (vgl. Abschnitt 3.2) und stellt eine Schnittstelle zum einheitlichen Nachrichtentransport über unterschiedliche Netzwerktechnologien bereit. Dies ist notwendig, um die bestehende Heterogenität einer mobilen Umgebung zu überwinden und die Geräte der mobilen Umgebung zu entkoppeln. Außerdem werden so eine möglichst große Anzahl von Systemen zunächst isolierter mobiler Umgebungen verbunden, sodass ihre Ressourcen für das neu

[2] http://vsis-www.informatik.uni-hamburg.de/projects/demac/

[3] Die virtuelle Maschine J9 ist die Umsetzung der Java-Plattform, Micro Edition, von IBM.

entstehende Gesamtsystem nutzbar werden. Hierbei bietet der *asynchrone Transportdienst* durch seine nachrichtenorientierte Kommunikation zwischen den einzelnen Systemen einer (mobilen) Umgebung eine besonders gute Basis, um hierauf erweiterte Kommunikations- und Middleware-Dienste sowie Anwendungen aufzubauen. Diese Basiskommunikation wird dann durch den *Ereignisdienst* um eine Komponente zur ereignisgesteuerten und somit proaktiven Kommunikation kompletiert. Insgesamt ist es so mit Hilfe des Transportsystems der Middleware möglich, sowohl Nachrichten gezielt zwischen zwei Systemen auszutauschen als auch Informationen an interessierte Systeme automatisch zu distribuieren. Damit baut die Systemplattform auf technologischen Ansätzen auf, die sich im Umfeld mobiler Systeme bewährt haben und somit eine vorteilhafte Grundlage für die erweiterten Dienste der Middleware bilden.

6.1.1. Nachrichtenorientierter Transportdienst

Der *asynchrone Transportdienst* der Systemplattform dient zum technologieneutralen Versand und Empfang von asynchronen Nachrichten innerhalb der (mobilen) Systemumgebung. Hierzu erzeugt der Transportdienst ein *Overlay-Netzwerk* (vgl. [DO03]), also eine virtuelle Netzwerktopologie, die über die verschiedenen physikalischen Netzwerktechnologien einer Systemumgebung gelegt wird (vgl. Abbildung 6.2). Dies ist notwendig, um die in mobilen Systemen vorhandenen unterschiedlichen Netzwerktechnologien miteinander zu verbinden und so die Möglichkeit zu gewinnen, Geräte zu einem Gesamtsystem zu verbinden, die aufgrund der unterschiedlichen Basistechnologien zuvor nicht miteinander interagieren konnten. Damit wird die Verteilungstransparenz in der mobilen Umgebung insgesamt größer und erweiterte Dienste können losgelöst von physikalischen Technologien und Protokollen konzipiert werden. Damit steigt zugleich die Wiederverwendbarkeit und Unabhängigkeit der Systemplattform zur Ausführung Mobiler Prozesse. Dabei muss auch bei der Konzeption dieses Diensts ein Ansatz gewählt werden, der durch eine strikte Komponentenorientierung und eine lose Kopplung eine dynamisch anpassbare und damit flexible Architektur bietet. Denn nur so kann den intrinsischen Eigenschaften mobiler Systeme (vgl. Abschnitt 2.4) Rechnung getragen werden und eine möglichst optimal an die Bedürfnisse und Fähigkeiten des lokalen Geräts angepasste Middleware konzipiert werden.

In diesem von der Transportschicht gebildeten virtuellen Netzwerk sind zunächst nur *Punkt-zu-Punkt-Verbindungen* möglich, die somit lediglich

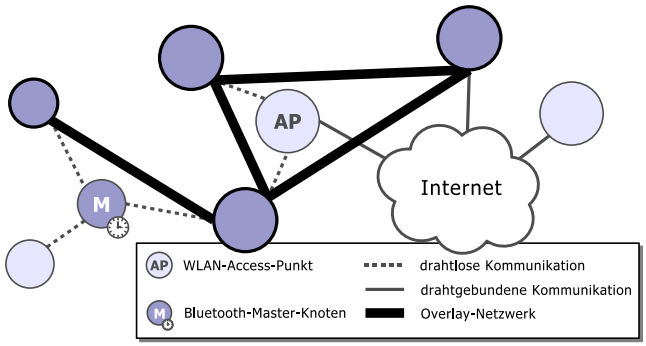


Abbildung 6.2.: Overlay-Netzwerk über verschiedene Netztechnologien

einen Nachrichtenaustausch zwischen zwei physikalisch direkt miteinander verbundenen Systemen bieten. Um auch indirekt verbundene Systeme im Overlay-Netzwerk zu erreichen, werden die physikalischen Verbindungen mehrerer Systeme zwischen Sender und Empfänger einer Nachricht genutzt, um eine *Multi-Hop-Kommunikation* und somit ein Nachrichten-Routing im virtuellen Netz zu erreichen. Dabei wird durch das Routing auf der Ebene der Middleware-Schicht bewirkt, dass Nachrichten auch über unterschiedliche Netzwerktechnologien hinweg versendet werden können und so die Netzwerkheterogenität für die auf dem Transportdienst aufbauenden Dienste und für dejenigen Anwendungen, die Middleware nutzen, verborgen wird (*Verteilungstransparenz*).

Zur Adressierung im Overlay-Netzwerk können prinzipiell alle eindeutigen Adressierungsschemata verwendet werden, die sogar parallel in einem System eingesetzt werden können, wenn sie disjunkte Namensräume aufspannen. In der prototypischen Implementierung werden hierfür weltweit eindeutige Bezeichner (UUID[4]) eingesetzt, die auch ohne zentrale Koordinierungsinstanz erzeugt werden können und so für eine unzuverlässige mobile Umgebung mit gleichberechtigten Teilnehmern besonders geeignet sind.

Um die Skalierbarkeit und Offenheit der Systemplattform gewährleisten zu können, muss das Overlay-Netzwerk möglichst autonom aufgebaut und administriert werden. Deshalb ist es notwendig, dass der asynchrone Transportdienst neben dem Nachrichtentransport auch Mechanismen beinhaltet, mit denen er erkennen kann, dass neue Systeme in das virtuelle Netzwerk eintreten und dass sie dieses gegebenenfalls wieder ver-

[4] UUIDs sind 128-Bit-lange Zahlen, deren Erzeugungsschema so definiert ist, dass ein Generieren zweier gleicher Bezeichner so unwahrscheinlich ist, dass eine UUID im alltäglichen Gebrauch als eindeutig angesehen werden kann.
(vgl. ISO/IEC-Standard 9834-8 [II05])

lassen haben. Damit diese Informationen leicht verwaltet werden können, müssen diese Informationen über erreichbare Teilnehmer strukturiert und einheitlich aufbereitet werden sowie durch den Einsatz von internen Verzeichnissen leicht zugreifbar gemacht werden. So ist dann zum Beispiel die Abbildung von logischen Adressen des Overlay-Netzwerkes zu physikalischen Adressen der darunterliegenden Transportmechanismen und -protokollen effizient möglich.

6.1.1.1. Kapselung physikalischer Netzwerke

Der Zugriff auf die physikalischen Netzwerke, wie zum Beispiel WLAN (vgl. Abschnitt 2.5.3.2) oder Bluetooth (vgl. Abschnitt 2.5.3.1), wird durch eine einheitliche Schnittstelle nach dem *Adapter-Strukturmuster* [GHJV03] gekapselt, die somit eine einheitliche Zugangsschicht zu den technologieabhängigen Transportprotokollen darstellt (vgl. Abbildung 6.1). Dies ermöglicht es, die Systemplattform an die Fähigkeiten des jeweiligen (mobilen) Gerätes anzupassen und so die Heterogenität bezüglich der verfügbaren Netzwerktechnologien zu überwinden. Hierzu werden die Adapter für die auf dem Gerät vorhandenen Netzwerkprotokolle während der Initialisierungsphase der Middleware-Plattform – zum Beispiel aus einer Konfigurationsdatei – eingelesen und dynamisch in das System geladen (*Plug-In-Mechanismus*).

Da zu übertragende Nachrichten, die an den asynchronen Transportdienst übergeben werden, lediglich an die Overlay-Netzwerkadresse gerichtet sind, muss die Transportschicht vor der Weiterleitung der Nachricht anhand seines internen Verzeichnisses eine physikalische Adresse des Kommunikationspartners und den entsprechenden Adapter zum Transport der Nachricht bestimmen. Auf diesen Vorgang der Adressauflösung kann dabei durch die Angabe von nicht-funktionalen Randbedingungen (`Constraints`) Einfluss genommen werden. Damit ist es dem System zum Beispiel möglich, Forderungen nach einer minimalen Bandbreite bei der Übertragung einer Nachricht durch die gezielte Auswahl von passenden Netzwerktechnologien durchzusetzen. Werden keine nichtfunktionalen Rahmenbedingungen gesetzt, wird aus allen verfügbaren Netzwerktechnologien, die eine Kommunikation zwischen dem sendenden und dem empfangenden System ermöglichen, eine Technologie nichtdeterministisch ausgewählt.

Neben dem gezielten Versand einzelner Nachrichten an einen Empfänger bietet der Transportdienst auch die Möglichkeit, Nachrichten an meh-

rere Empfänger gleichzeitig zu senden (*Multicast-Nachrichten*). Dabei können solche Multicast-Nachrichten mit unterschiedlichen Reichweiten versendet werden: Zum einen kann die Reichweite auf die unmittelbare Peripherie des Senders beschränkt werden, was auf die Menge der Systeme abgebildet wird, die unmittelbar über physikalische Verbindungen mit dem Sender verbunden sind. Zum anderen kann eine Nachricht aber auch an alle dem lokalen System bekannten Empfänger verschickt werden, indem eine globale Reichweite angegeben wird.

Um die asynchrone Kommunikation an andere Komponenten des Systems anzubinden, muss ein Mechanismus eingesetzt werden, der die nicht-deterministisch eintreffenden Kommunikationsnachrichten so an die Empfangskomponenten weiterreicht, dass diese sinnvoll auf die Nachrichten reagieren können. Zur Umsetzung dieser Forderung bietet sich das *Beobachter-Verhaltensmuster* [GHJV03] als besonders geeignete Methode an. Hierbei registrieren sich diejenigen Systemkomponenten, die Nachrichten empfangen möchten, beim Transportdienst und werden beim Eintreffen von entsprechenden Nachrichten durch den Aufruf einer *Call-Back-Methode* über deren Eingang informiert. Hiermit wird eine besonders lose Kopplung der Komponenten erreicht, da keine Annahmen vom Transportdienst über die Anzahl und die Funktionalität der Empfangskomponenten gemacht werden, sodass die Empfänger nicht vorab bekannt sein müssen. Insgesamt entsteht so eine besonders flexible Kommunikation, an der unterschiedliche Teilsysteme dynamisch teilnehmen können. Um bei dieser asynchronen Benachrichtigungsmethode Nachrichten einzelnen Empfängern zuordnen zu können, müssen die Nachrichten einen Bezeichner enthalten, der die Unterscheidung verschiedener Nachrichtentypen ermöglicht.

Die Nutzdaten einer Nachricht bestehen aus einer beliebig langen Zeichenkette, die zusätzlich zum Nachrichtenkörper die Adresse des Senders und des Empfängers sowie einen Nachrichtenbezeichner in einem XML-kodierten Dokument zusammenfasst, um eine plattformunabhängige Beschreibung der Gesamtnachricht zu erhalten. Um Binärdaten mit Hilfe einer solchen zeichenorientierten Übertragungsmethode zu versenden, müssen diese zuvor, zum Beispiel durch das von der Internet Society standardisierte *Base64-Verfahren* [Jos06], in eine entsprechende Zeichenkette umgewandelt werden – was jedoch die allgemeine Verwendbarkeit des Ansatzes nicht einschränkt.

6.1.1.2. Auffinden von Kommunikationspartnern

Um das Overlay-Netzwerk autonom aufbauen und administrieren zu können, ist es erforderlich, dass der asynchrone Transportdienst Systeme, die in die mobile Umgebung eintreten, finden kann und erkennt, wenn ein bereits bekanntes Gerät die Umgebung wieder verlässt. Um dies zu erreichen, muss die Suche nach neuen Kommunikationspartnern auf die einzelnen Netzwerktechnologien abgebildet werden. Deshalb bietet die Adapterschnittstelle die Möglichkeit, technologiespezifisch nach neuen Systemen zu suchen, die ebenfalls über die Systemplattform zur Umsetzung Mobiler Prozesse verfügen. Die so gewonnenen Informationen werden dann in einem transportdienstinternen Verzeichnis verwaltet (vgl. Abbildung 6.3). Hierbei werden bekannte Geräte als `Device`-Objekte abgebildet, die alle

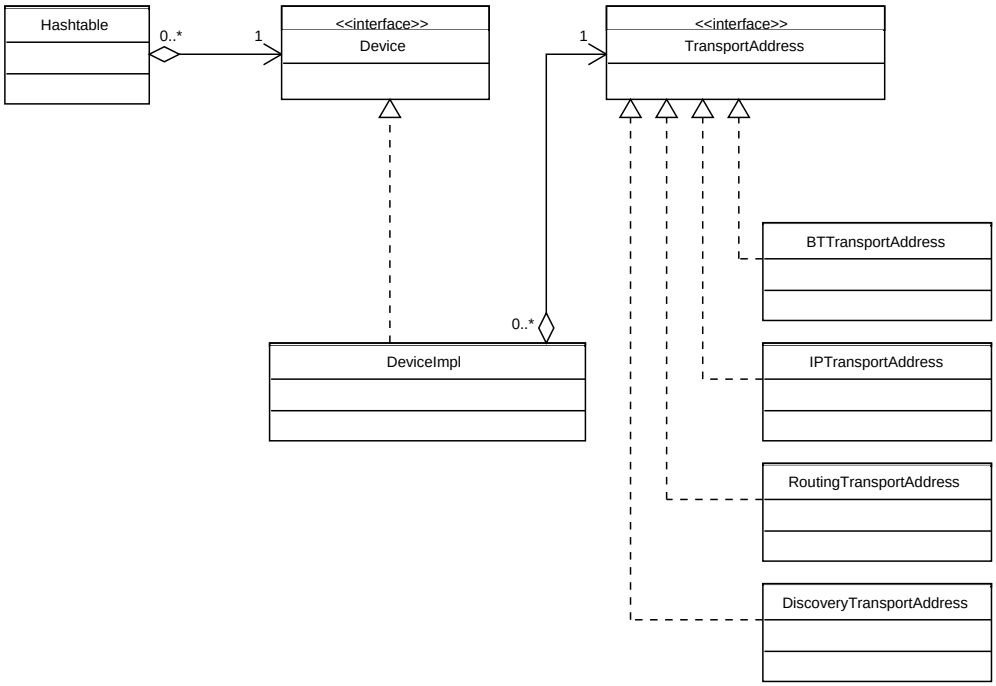


Abbildung 6.3.: Verzeichnis zur Verwaltung des Overlay-Netzwerk

Informationen über das jeweilige entfernte Gerät speichern. Dabei aggregieren sie vor allem auch alle physikalischen beziehungsweise virtuellen Netzwerkadressen, die zu dem Gerät bekannt sind. Die so erkannten Systeme werden dann in einem `Hashtable`-Objekt als Basisstruktur des Verzeichnisses unter ihrem eindeutigen Bezeichner im Overlay-Netzwerk abgespeichert. Kann ein Gerät über eine gespeicherte Netzwerkadresse nicht mehr erreicht werden, wird diese aus der Liste der verfügbaren Adressen wieder gestrichen, sodass nur aktive Verbindungen gespeichert bleiben.

Die technologiespezifische Suche muss dabei individuell auf die Gegebenheiten der einzelnen Netzwerkprotokolle zugeschnitten werden. So wird die Suche in IP-basierten Netzwerken zum Beispiel über eine Multicast-Anfrage realisiert. Hierbei wird eine Nachricht über das Netzwerk an eine lokale Multicast-Gruppe gesendet, der alle Plattformen zur Ausführung von Mobilen Prozessen beitreten müssen, um über diesen Mechanismus gefunden zu werden. Bei einem Bluetooth-Netzwerk hingegen, registriert sich die Plattform direkt beim Bluetooth-System als Dienst und kann dann über dieses mit Hilfe des im Protokollstapel vorgegebenen *Service Discovery Protocol* (vgl. Abschnitt 2.5.3.1) gefunden werden. Alle so entdeckten Geräte werden zusammen mit den zugehörigen technologiespezifischen Adressen entsprechend dem Beobachter-Verhaltensmuster asynchron an den Transportdienst gemeldet, der mit diesen Informationen sein internes Verzeichnis aktualisieren kann.

Wird nun eine Nachricht an den asynchronen Transportdienst übergeben, so sucht dieser zunächst das zum Empfänger gehörende Gerät aus dem Verzeichnis heraus und wählt dann, gegebenenfalls eingeschränkt durch nicht-funktionale Randbedingungen, eine der Transportadressen aus. Aufgrund dieser Entscheidung wird dann der zur Transportadresse passende Adapter verwendet, um die Nachricht an den Empfänger zu versenden.

Um die Informationen im Verzeichnis aktuell zu halten, ist jeder Eintrag über ein entferntes Gerät mit einer Lebenszeit versehen (*Lease Time*). Läuft diese Lebenszeit ab, fragt das lokale Gerät beim entfernten an, ob dieses noch erreichbar ist. Erhält das lokale System innerhalb einer parametrisierbaren Zeitspanne eine Quittungsnachricht, wird die Lebenszeit des entfernten Gerätes verlängert. Trifft keine Quittungsnachricht innerhalb der geforderten Zeitspanne ein, wird das entfernte Gerät aus der Liste der erreichbaren Geräte wieder gelöscht und somit der lokale Datenbestand aktualisiert. Trifft die Quittungsnachricht verspätet ein, wird das gelöschte Gerät wieder im Verzeichnis registriert, sodass nur endgültig nicht erreichbare Geräte dauerhaft gelöscht werden.

6.1.1.3. Virtuelle Netzkomponenten

Alle Netzwerkoperationen, die sich auf das Overlay-Netzwerk und nicht auf die darunterliegenden physikalischen Verbindungen beziehen, werden von so genannten virtuellen Netzkomponenten ausgeführt. Diese Komponenten besitzen dabei meist die gleiche Adapterschnittstelle wie die Zu-

griffskomponenten, mit denen die physikalischen Kommunikationsnetzwerke angesprochen werden (vgl. Abbildung 6.1). Zu den virtuellen Netzkomponenten zählen das *Routing im Overlay-Netzwerk*, die *Kommunikation mit unbekannten Adressaten* und die Komponente zum Versand *verschlüsselter Nachrichten*.

Routing im Overlay-Netzwerk Durch das Bilden des Overlay-Netzwerkes wird nicht nur eine Reduktion der Heterogenität im mobilen System angestrebt, sondern auch eine Erweiterung der Kommunikation über unterschiedliche Netzwerktechnologien hinweg. Dazu müssen jedoch Nachrichten auch zwischen Systemen ausgetauscht werden können, die nicht direkt miteinander verbunden sind. Deshalb muss in die Kommunikationsschicht der Systemplattform ein Mechanismus integriert werden, der Nachrichten auch indirekt mit Hilfe von vermittelnden Systemknoten realisiert (*Nachrichten-Routing*). Hierzu müssen Nachrichten, die zwischen nur indirekt verbundenen Geräten ausgetauscht werden sollen, so vom Nachrichtentransportsystem behandelt werden, dass sie Schritt für Schritt über mehrere physikalische Verbindungen hinweg zum eigentlichen Empfänger geleitet werden. Hierzu wird in der Systemplattform ein so genannter virtueller Netzwerkadapter eingesetzt, der das Routing im Overlay-Netzwerk realisiert. Dabei werden Nachrichten, die nicht direkt dem Empfänger zugestellt werden können, in Nachrichten an vermittelnde Konten (*Mediatoren*) umgewandelt, die dann an die entsprechenden Netzwerkknoten zwischen Sender und Empfänger der ursprünglichen Nachricht zur weiteren Bearbeitung weitergeleitet werden. Hierbei nutzt die Routing-Komponente die gleiche Methode zur Auswahl des Transportmechanismus wie auch die Adapter zur Kapselung technologieabhängiger Transportprotokolle. Deshalb werden die Routing-Informationen als Transportadressen gekapselt (`RoutingTransportAddress`) und im internen Verzeichnis des Transportdiensts gespeichert. Wird nun eine Nachricht an den asynchronen Transportdienst übergeben, dessen Empfänger nicht direkt erreichbar ist, wird als Transportadresse die Routing-Information ausgelesen und der *Routing-Adapter* für den Versand verwendet. Dieser erzeugt dabei eine neue Nachricht, die als Routing-Nachricht im Nachrichtenbezeichner gekennzeichnet ist und setzt den Mediator als neuen Zielknoten sowie die alte Nachricht als neuen Inhalt ein. Diese Nachricht braucht dann nur dem asynchronen Transportdienst zum Versand übergeben werden, der sie wie eine gewöhnliche Nachricht zustellen kann. Auf der Seite des Empfängers werden aus den als Routing-Nachrichten gekennzeichneten

Nachrichten die ursprünglichen Daten wieder ausgelesen und je nach dem, ob die enthaltene Nachricht für das lokale System bestimmt ist oder ob die Nachricht weitergeleitet werden muss, unterschiedlich behandelt: Ist die gekapselte Nachricht an das lokale System gerichtet, kann sie normal an die als Beobachter registrierten Systemkomponenten ausgeliefert werden; ist sie es jedoch nicht, wird die nächste Transportadresse bestimmt und die Nachricht als neue Routing-Nachricht über den entsprechenden Adapter weitergereicht. Somit wandern die ursprünglichen Daten Schritt für Schritt und gegebenenfalls über verschiedene Netzwerktechnologien hinweg zu ihrem endgültigen Empfänger. Dabei werden die notwendigen Routing-Informationen jedoch nicht direkt vom Routing-Adapter bereitgestellt, sondern durch eine andere virtuelle Netzwerkkomponente ermittelt – dem *Discovery-Adapter*, dessen Hauptaufgabe darin liegt, Nachrichten an Adressaten zu senden, die im internen Verzeichnis (noch) nicht zu finden sind.

Kommunikation mit unbekannten Adressaten Unter anderem bei der Ausführung von Mobilen Prozessen kann die Situation eintreten, dass Nachrichten an Adressaten versendet werden sollen, die dem asynchronen Transportdienst nicht bekannt sind. Dies ist zum Beispiel dann der Fall, wenn im Mobilen Prozess ein spezielles Gerät als Ausführungsort einer Aktivität definiert ist. In diesen Situationen muss vom Transportdienst versucht werden, den Kommunikationspartner in der mobilen Umgebung zu finden. Hierfür ist in der Systemplattform der *Discovery-Adapter* vorgesehen, der die zunächst unzustellbaren Nachrichten zwischenspeichert, eine Suche nach den Adressaten initiiert und Nachrichten, deren Adressat gefunden wurde, an diesen weiterleitet.

Da die Suche nach unbekannten Adressaten ebenfalls als Netzwerkadapter konzipiert ist, wird, sollte nach der Suche einer physikalischen beziehungsweise indirekten Verbindung zum Empfänger keine geeignete Transportadresse vorliegen, eine spezielle Transportadresse generiert (`DiscoveryTransportAddress`), die den Transportdienst dazu veranlasst, den Discovery-Adapter als Transportkomponente zu wählen. Dieser stellt die Nachricht dann jedoch zunächst in einen dem Empfänger zugeordneten Nachrichtenpuffer und löst eine Suche nach einer Netzwerkadresse aus, die zum Adressaten führt. Dies wird erreicht, indem der Adapter eine Anfrage (`RouteQuery`) an alle bekannten Geräte sendet. Dabei sind Query-Nachrichten so konzipiert, dass sie über mehrere Knoten weitergereicht werden (*Multi-Hop-Nachrichten*), bis eine maximale Anzahl von Zwischen-

schritten erreicht wurde (*Time-to-Live-Ansatz*).

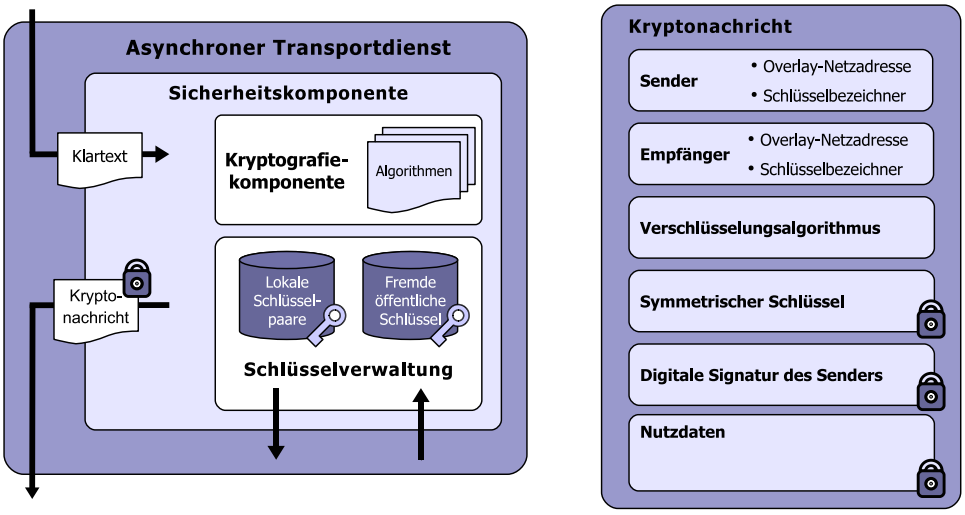
Erreicht eine solche Anfrage einen Mediator, der die Nachricht weiterleiten kann, da er eine physikalische oder virtuelle Verbindung zum gesuchten Knoten kennt, sendet dieser in einer Antwort (`RouteResponse`) alle Angaben, die zum Erzeugen einer entsprechenden Routing-Adresse benötigt werden, an den suchenden Knoten. Mit dem Eintreffen der Routing-Informationen wird ein entsprechender Eintrag im internen Verzeichnis des Transportdiensts eingefügt und der Discovery-Adapter – per Beobachter-Verhaltensmuster – über den Eingang der Informationen informiert. Daraufhin wird der Nachrichtenpuffer vom Discovery-Adapter aufgelöst, indem alle bis dahin aufgelaufenen Nachrichten wieder dem Transportdienst zur Übermittlung übergeben werden.

Da es aber vorkommen kann, dass auch so kein Weg zum Adressaten gefunden wird, müssen Nachrichten, die nicht zugestellt werden können, nach Ablauf einer vorgegebenen Zeitspanne (*Timer*) wieder aus dem System gelöscht werden, um die Ressourcen des mobilen Systems zu schonen. Hierzu wird jede Nachricht in der Warteschlange mit einem Zeitstempel versehen und alle Warteschlangen werden regelmäßig auf abgelaufene Timer überprüft und dabei die obsoleten Nachrichten gelöscht.

Um die lokalen Informationen über das Overlay-Netzwerk sukzessive zu erweitern, ist es über den Discovery-Adapter auch möglich, die den anderen Teilnehmern bekannten Systeme der mobilen Umgebung abzufragen. Hierzu wird eine entsprechende Anfrage nach der Liste der bekannten Geräte (`DeviceListQuery`) an ein entferntes System gesendet, das dann die entsprechende Liste liefern kann.

Sicherheitsaspekte auf Ebene des Transportdiensts Sicherheitsaspekte sind im Umfeld Mobiler Prozesse besonders wichtig, da sie eine vertrauensvolle Kooperation zwischen unterschiedlichen Teilnehmern einer mobilen Umgebung erst ermöglichen (vgl. Abschnitt 5.4.6). Deshalb müssen auf allen Ebenen einer Systemplattform adäquate Sicherheitsmechanismen integriert werden. Auf der Ebene des Nachrichtentransportes bedeutet dies, dass eine sichere Kommunikation zwischen Sender und Empfänger bereitgestellt werden muss. Dabei ist das gewählte Sicherheitsverfahren so zu konzipieren, dass Mediatorknoten zum einen nicht auf die Inhalte der Nachricht im Klartext zugreifen können und dass sie zum anderen nicht zwingend selbst Sicherheitskomponenten bereithalten müssen, um Nachrichten zwischen den kommunizierenden Knoten zu vermitteln. Aus diesem Grund stellt der asynchrone Transportdienst im Bedarfsfall ei-

ne Sicherheitskomponente zur verschlüsselten Kommunikation zwischen zwei Systemen im Overlay-Netzwerk bereit (vgl. Abbildung 6.4(a) und [Win07]). Mit dieser Komponente wird erreicht, dass die Systemplattform unabhängig von den verfügbaren Netzwerktechnologien ist und dabei auch über Zwischenknoten hinweg die Vertraulichkeit von Nachrichten gewährleistet werden kann. Dabei ist diese so konzipiert, dass, wie gefordert, leistungsschwächere Zwischenknoten des Systems auch auf die Sicherheitskomponente verzichten können, ohne dabei als Mediator für kryptografisch geschützte Nachrichten auszufallen. Um dies zu realisieren, muss diese Komponente also so konzipiert sein, dass letztendlich alle kommunizierten Nachrichten der Standardstruktur für Nachrichten des asynchronen Transportdiensts entsprechen.



(a) Sicherheitskomponente des Transportdienstes

(b) Struktureller Aufbau einer Kryptonachricht

Abbildung 6.4.: Integration verschlüsselter Kommunikation

Um Nachrichten über den asynchronen Transportdienst verschlüsselt übertragen zu können, verwenden Klienten sichere Transportnachrichten (`SecureTransportMessage`), welche die Angaben normaler Nachrichten (`TransportMessage`) um Informationen über die gewünschten Verschlüsselungsalgorithmen als nicht-funktionale Parameter ergänzen. Ist kein spezieller Algorithmus angegeben, muss vom Transportdienst ein von ihm unterstützter Algorithmus gewählt werden, denn es ist ihm unter keinen Umständen erlaubt, eine sichere Nachricht unverschlüsselt zu senden. Eine solche sichere Transportnachricht kann aufgrund der Vererbungsbeziehung über die gleiche Schnittstelle an den Transportdienst

übergeben werden, die auch für unverschlüsselte Nachrichten verwendet wird. Der Transportdienst übergibt die zu verschlüsselnde Transportnachricht vor dem Senden über die Netzwerkadapter jedoch zunächst an die von ihm gekapselte Sicherheitskomponente, die aus der Klartextnachricht eine Kryptonachricht macht, die dann als Nutzdaten einer normalen Transportnachricht versendet wird (vgl. Abbildung 6.4(a)). Dabei wird zum Erstellen der Kryptonachricht ein hybrides Verschlüsselungsverfahren angewendet; dies bedeutet, dass sowohl symmetrische als auch asymmetrische Verschlüsselungsalgorithmen eingesetzt werden. Das bietet den Vorteil, dass nur der für diese Nachricht verwendete symmetrische Schlüssel durch das komplexer zu berechnende asymmetrischen Public-Key-Verfahren verschlüsselt werden muss. Damit wird erreicht, dass insgesamt die oft begrenzten Ressourcen der mobilen Kommunikationspartner bei der Verschlüsselung umfangreicher Nutzdaten geschont werden können. In der prototypischen Implementierung der Systemplattform im Rahmen des Projekts DEMAC werden hierfür als symmetrische Verschlüsselungsmethode der *Advanced Encryption Standard* (AES) und als Public-Key-Kryptosystem das RSA[5]-Verfahren verwendet. Dabei ist die Architektur der Sicherheitskomponente so zu wählen, dass sie nicht auf diese bestimmten Verfahren zur Verschlüsselung der Nachrichten festgelegt ist, sondern dass die verwendeten Verfahren durch die Angabe von nicht-funktionalen Parametern bestimmt werden können. Dadurch wird eine höhere Flexibilität erreicht, sodass Anwendungen und Dienste, die diese Komponente nutzen, ihre eigenen Sicherheitsstandards definieren können. Zudem ist die Systemplattform so leicht mit neu entwickelten kryptografischen Verfahren erweiterbar und bleibt somit auch unter zukünftigen Rahmenbedingungen einsetzbar.

Aufgrund der Möglichkeit, in der erweiterten Transportnachricht einen Verschlüsselungsalgorithmus durch die Angabe von nicht-funktionalen Parametern anzugeben, muss auch die Anbindung der Verschlüsselungsverfahren so konzipiert sein, dass sie in eigenen austauschbaren Komponenten gekapselt sind. Deshalb werden die einzelnen Verfahren in eigenständigen Kryptosystemen implementiert, die von der Sicherheitskomponente entsprechend der Angaben in der sicheren Transportnachricht gewählt und dynamisch eingebunden werden. Nach dieser dynamischen Konfigurierung der Sicherheitskomponente wird dann der zur Kodierung der Nutzdaten zu verwendende symmetrische Schlüssel lokal erzeugt und die Nutz-

[5] RSA ist ein Akronym der Namen der Entwickler Ronald L. Rivest, Adi Shamir und Leonard Adleman.

daten mit diesem verschlüsselt. Über die nun kryptografisch gesicherten Daten wird zudem noch ein Hash-Wert als *Message Digest* gebildet und mit dem privaten Schlüssel des lokalen Systems verschlüsselt, sodass eine digitale Signatur der Nutzdaten entsteht. In der prototypischen Systemimplementierung wird hierzu das SHA-1-Hash-Verfahren verwendet. Zudem wird der symmetrische Schlüssel, der zur Chiffrierung der Nutzdaten verwendet wurde, mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Sollte dieser noch nicht lokal vorliegen, wird er durch ein Request-Reply-Protokoll zunächst vom Empfänger angefordert. Die so erzeugten Daten werden dann in einer *Kryptonachricht* zusammengefasst (vgl. Abbildung 6.4(b)) und als Nutzdaten einer regulären und als chiffriert gekennzeichneten Transportnachricht an den asynchronen Transportdienst zurückgegeben, der sie dann normal weiterverarbeiten kann. Die Kryptonachricht enthält dabei Angaben zum Sender und Empfänger der Nachricht jeweils in Form der eindeutigen Overlay-Netzadresse und Informationen zum in der Nachricht verwendeten öffentlichen Schlüssel, damit der Empfänger den richtigen privaten Schlüssel zum dechiffrieren des symmetrischen Schlüssels finden kann. Des Weiteren enthält die Kryptonachricht Angaben zum verwendeten Verschlüsselungsalgorithmus, den verschlüsselten symmetrischen Schlüssel, die digitale Signatur sowie die chiffrierten Nutzdaten.

Auf Empfängerseite werden alle als chiffriert gekennzeichneten Transportnachrichten nicht sofort an die Beobachter weitergereicht, vielmehr wird die enthaltene Kryptonachricht zunächst entnommen und zum Dechiffrieren an die Sicherheitskomponente gegeben. Diese überprüft zunächst die digitale Signatur der Nachricht und dechiffriert dann mit ihrem privaten Schlüssel den symmetrischen Schlüssel der Kryptonachricht. Anschließend werden die verschlüsselten Nutzdaten mit dem entschlüsselten symmetrischen Schlüssel dechiffriert und eine neue reguläre Transportnachricht erzeugt, welche die nun wieder im Klartext vorliegenden Daten enthält. Diese wird dann wieder an den Transportdienst zur Distribution zurückgegeben.

6.1.2. Ereignisbasierte Erweiterung des Transportdiensts

Um in einer mobilen Umgebung automatisch über Veränderungen der Umgebung informiert zu werden, sind proaktive Kommunikationsformen besonders wichtig. In einer Systemplattform zur Realisierung Mobiler Prozesse können so zum Beispiel Kontextdaten besonders einfach zwischen

unterschiedlichen Geräten synchronisiert werden. Aus diesem Grund muss auch das hier vorgestellte Middleware-Konzept proaktive Kommunikationsformen integrieren. Deshalb bietet die prototypische Implementierung einen eigenständigen *Ereignisdienst*, der auf dem Transportdienst aufsetzt und dessen Dienstleistung zum asynchronen Nachrichtenversand nutzt, um eine Kommunikation entsprechend dem *Publish/Subscribe-Modell* (vgl. Abschnitt 3.2.2) zu realisieren.

6.1.2.1. Verwalten von Ereignissen

Der Ereignisdienst der Systemplattform hat als primäre Aufgaben die Verwaltung von lokalen und entfernten Abonnenten, die sich für Ereignisse interessieren, sowie die Verteilung von Ereignisnachrichten an eben diese Abonnenten. Dies bedeutet, dass – mit Ausnahme von wenigen Systemereignissen – die Quellen der Ereignisse außerhalb des Ereignisdiensts angesiedelt sind. Dies können weitere Systemkomponenten sein wie der Kontext- oder Prozessdienst (vgl. Abbildung 6.1) oder aber Ereignisse, die aus Anwendungen heraus entstehen. Die bereits vom Ereignisdienst bereitgestellten Ereignisse sind die Anzeige von neu erkannten Geräten (`DeviceDetectedEvent`) und die Information über beim Ereignisdienst eingehende Abonnements (`RegisterEvent`). Alle Ereignisse, die vom System oder von Anwendungen generiert werden, können zwar lokal ohne vorherige Registrierung des Ereignistyps verwendet werden, sollen jedoch die Ereignisse von externen Quellen bezogen werden, ist eine Registrierung des entsprechenden Typs sinnvoll. Denn durch das Registrieren des Typs können die zugehörigen Ereignisnachrichten bereits vom Ereignisdienst in entsprechende Ereignisobjekte umgewandelt werden und so Anwendungen und Dienste der Systemplattform vom Transport der Ereignisse im Overlay-Netzwerk entkoppelt werden. Wird diese Registrierung nicht vorgenommen, werden die empfangenen Ereignisnachrichten in einem generischen Ereignistyp gekapselt (`UnknownEvent`) und ausgeliefert. Eine gegebenenfalls notwendige Umsetzung in ein entsprechendes Ereignisobjekt muss dann jedoch vom Abonnenten selbst vorgenommen werden.

Zur Verwaltung der Abonnements enthält der Ereignisdienst zwei Verzeichnisse: eines für lokale Abonnements und eines für die von entfernten Systemen. Diese Verzeichnisse verwalten für jeden Ereignistyp, der durch einen eindeutigen Bezeichner, zum Beispiel einer UUID, gekennzeichnet ist, eine Liste von Abonnenten. Wird dem Ereignisdienst, zum Beispiel von einer Anwendung, ein Ereignis (`Event`) zur Distribution übergeben,

sucht sich der Dienst je nach angegebener Reichweite des Ereignisses (lokal, peripher oder global) die passenden Abonnenten heraus und informiert diese über dessen Eintritt. Lokale Abonnenten werden dabei nach dem Beobachter-Verhaltensmuster benachrichtigt.

Um dem Ziel gerecht zu werden, mit Netzwerkressourcen schonend umzugehen, wird das Registrieren der Benachrichtigungswünsche auf entfernten Systemen zweistufig vorgenommen. Dabei muss eine Bündelung der auszutauschenden Ereignisnachrichten erfolgen, sodass die Anzahl der auszutauschenden Nachrichten minimiert wird. Deshalb melden sich entfernte Komponenten zunächst mit ihrem Abonnementwunsch auch lokal bei ihrem Ereignisdienst an, geben aber zusätzlich die Adresse des gewünschten Systems im Overlay-Netzwerk als zusätzlichen Parameter an. Daraufhin registriert der lokale Ereignisdienst den Abonnenten zunächst in seinem eigenen Verzeichnis und veranlasst dann den Ereignisdienst auf dem entfernten Gerät dazu, ihn selbst als Abonnenten des Ereignisses zu registrieren. Damit wird erreicht, dass die geforderte Bündelung der kommunizierten Ereignisnachrichten stattfindet und nur eine Ereignisnachricht pro System über das Netzwerk übertragen werden muss – auch dann, wenn es auf einem Gerät mehrere Abonnenten für einen Ereignistyp eines entfernten Gerätes gibt. Dies führt also dazu, dass die Netzwerkressourcen effizient genutzt werden und somit die zentrale Anforderung an eine mobile Systemplattform nach einem sparsamen Ressourceneinsatz und einer möglichst effektiven Nutzung der Kommunikationsverbindungen bei der Umsetzung berücksichtigt wird. Erhält daraufhin ein entferntes System eine Ereignisnachricht über den asynchronen Transportdienst, wird diese vom lokalen Ereignisdienst zunächst interpretiert, dann in ein entsprechendes lokales Ereignis transformiert und letztendlich an die Abonnenten ausgeliefert.

6.1.2.2. Integration individueller Ereignisse

Wird die ereignisbasierte Kommunikation auf Basis individueller Ereignisse zwischen entfernten Geräten benötigt, kann der Ereignisdienst durch das Registrieren von entsprechenden *Parsern*, die jeweils eine *Fabrikmethode* [GHJV03] für das Ereignis implementieren, so erweitert werden, dass er Ereignisnachrichten, die er über den asynchronen Transportdienst des Systems erhält, bereits als Ereignisobjekte an die Abonnenten distribuiert. Durch diesen Ansatz kann erreicht werden, dass die Abonnenten von den Details des Transports der Ereignisse entkoppelt werden, da die

Verarbeitung der serialisierten Ereignisdaten vollkommen vom Ereignisdienst gekapselt wird.

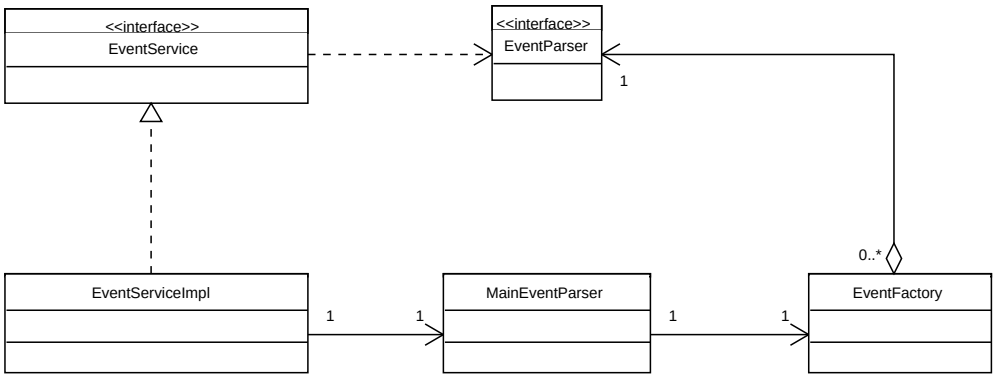


Abbildung 6.5.: Fabrik-Erzeugungsmuster zur Integration individueller Ereignisse

Um diese Kapselung des Erzeugens von Ereignisobjekten umzusetzen, wurde das Generieren der Objekte im Ereignisdienst entsprechend des *Fabrik-Erzeugungsmusters* [GHJV03] realisiert (vgl. Abbildung 6.5). Die benötigten Fabrikmethoden zum Erzeugen der Ereignisobjekte werden hierbei durch die Verwendung individueller Parser-Komponenten (`EventParser`) bereitgestellt, die vom Ereignisanbieter beziehungsweise -nutzer selbst geschrieben und beim Ereignisdienst über seine allgemeine Schnittstelle registriert werden müssen.

Treffen Ereignisnachrichten beim Ereignisdienst ein, extrahiert er die Daten, die zum Ereignis gehören, und übergibt diese an eine interne Verarbeitungskomponente (`MainEventParser`), welche die Daten vorverarbeitet und die eigentliche Fabrikkomponente (`EventFactory`) zum Erzeugen der Ereignisobjekte aufruft. Die Fabrik sucht dann anhand des Ereignistyps nach einem geeigneten Parser für das Ereignis, der die Syntaxanalyse der Daten durchführt und aus den so gewonnenen Strukturdaten die entsprechenden Ereignisobjekte generiert. Dieses Ergebnisobjekt wird dann an den Ereignisdienst zurückgegeben und von ihm an alle interessierten Abonnenten verteilt. Sollte kein passender Parser gefunden werden, wird von der Fabrik ein generisches Ereignisobjekt erzeugt, das über seine Schnittstelle Zugriff auf die allgemeinen Daten des Ereignisses ermöglicht, aber die eigentlichen Nutzdaten uninterpretiert lässt. So ist es selbst ohne geeigneten Parser häufig möglich, auf Ereignisse zu reagieren und es wird insgesamt die Robustheit des Systems gestärkt.

6.1.3. Fazit zur Transportschicht der Systemplattform

Durch das vom Kommunikationssystem etablierte Overlay-Netzwerk, das sowohl eine nachrichtenorientierte als auch eine ereignisbasierte Kommunikation zur Verfügung stellt, werden bereits viele grundlegende Eigenschaften, die eine Middleware für mobile Systeme bieten soll (vgl. Abschnitt 3.1 und Tabelle 3.1), erfüllt beziehungsweise es wird mit diesem System eine wesentliche Grundlage für deren Erfüllung gelegt. Dabei steht bei der Bildung des Overlay-Netzwerks vor allem das Überwinden der technologischen Heterogenität auf Ebene der Netzwerktechnologien im Vordergrund. Dies führt dazu, dass die (mobilen) Systeme Zugang zu möglichst vielen Geräten in der Umgebung und somit zu deren Ressourcen erhalten. So wird durch ein autonomes Netz-Management die Dynamik in mobilen Umgebungen berücksichtigt und somit eine offene Plattform geschaffen, in der neue Systeme automatisch integriert und bereits eingebundene Geräte dynamisch wieder entfernt werden können. Dabei wird durch Verwendung einheitlicher Protokolle und XML-basierter Kommunikationsnachrichten die Plattformunabhängigkeit des Gesamtsystems erreicht. Zudem wird es anderen Komponenten der Systemplattform durch die Integration von Sicherheitsmechanismen bereits auf der Ebene der Basiskommunikation ermöglicht, erweiterte Sicherheits- und Vertrauenskonzepte komfortabel umzusetzen.

Insgesamt wird mit dem Kommunikationssystem ein hoher Grad an Verteilungstransparenz und Fehlertoleranz erreicht, der jedoch genug Raum für Anpassungen des Systems an die Mobilität der einzelnen Geräte lässt. Dabei wird durch eine konsequente Komponentenbildung und die dynamische Konfiguration des Systems erreicht, dass sich das Kommunikationssystem an die Bedürfnisse und Fähigkeiten des jeweiligen (mobilen) Gerätes anpassen kann.

6.2. Der Kontextdienst

Die Umsetzung des in Abschnitt 5.6 eingeführten föderierten und generischen Kontextkonzepts erfolgt in der Systemplattform zur Realisierung der kontextbasierten Kooperation durch den so genannten ”´Kontextdienst“. Dabei besteht dieser Dienst selbst wiederum aus zwei miteinander gekoppelten Hauptkomponenten: Die erste Komponente dient zum Management von lokalen und entfernten Kontextdaten, die dann durch die zweite Komponente ergänzt wird, die für das Auflösen von abstrakten Dienstklassen in der mobilen Umgebung zuständig ist (vgl. Abschnitt

5.6.3). Insgesamt entsteht so eine Dienstkomponente für die Systemplattform, die alle kontextrelevanten Aktionen bündelt und unter einer einheitlichen Schnittstelle kapselt. Die einzelnen Sensoren und Dienste, die Kontextdaten liefern, werden hierbei durch so genannte Rohdatendienste (`RawDataServices`) nach dem *Adapter-Strukturmuster* [GHJV03] gekapselt, sodass sie in einheitlicher Form an die Verwaltungskomponente angebunden und den Attributen des Kontextmodells zugeordnet werden können.

6.2.1. Kontext-Management

Um anderen Diensten und Anwendungen der Systemplattform Kontextdaten einheitlich und effizient anbieten zu können, muss die Management-Komponente des Kontextdienstes sowohl das generische Kontextmodell (vgl. Abbildung 5.8) als auch die zu dessen Verwaltung notwendigen Subdienste bereitstellen. Dabei wird eine klare Trennung zwischen Modell und Verwaltungskomponenten angestrebt, um so die Kontextdaten über die Grenzen lokaler Plattformen und Programmiersprachen hinweg nutzen zu können (vgl. Abschnitt 5.6.1).

6.2.1.1. Umsetzung des verteilten Kontextmodells

Das generische Kontextmodell, das in Abschnitt 5.6 entwickelt wurde, kann semantisch in ein Domänensubmodell zum Abbilden von anwendungsabhängigen Ausschnitten aus dem Gesamtkontext und ein Attributmodell als einheitliches Abbild der einzelnen grundlegenden Kontextinformationen aufgeteilt werden. Verknüpft sind diese beiden Modellteile durch die in den Domänen zusammengefassten Entitäten, die Attribute zu Objekten der physikalischen und virtuellen Welt aggregieren und somit diese zu einer erweiterten semantischen Einheit verbinden.

Zugegriffen wird auf konkrete Kontextinformationen über Gerätekontexte (`DeviceContext`), die über den Kontextdienst bezogen werden können. Diese sind dabei, wie alle anderen Teile des generischen Kontextmodells, zunächst als einheitliche Schnittstellen definiert, die dann durch die unterschiedlichen Kontextdatenanbieter implementiert werden und zu einem einheitlichen Kontextmodell vereinigt werden (vgl. Abbildung 6.6). Die verfügbaren Gerätekontexte teilen sich hierbei in den Kontext des lokalen Gerätes (`LocalDeviceContext`) und in die Kontexte entfernter Geräte (`RemoteDeviceContext`) auf und werden über die eindeutigen Adressen der Systeme im Overlay-Netzwerk referenziert. Dabei kann nur

vom lokalen System aus und über die einzelnen Domänen auf den hiesigen Kontext Einfluss genommen werden. Dazu bietet der lokale Kontext zusätzlich zu den Standardmethoden aller Kontexte weitere Methoden an, mit denen Domänen zugegriffen, hinzugefügt und gelöscht werden können – nur bei entfernten Kontexten ist eine Modifikation nicht möglich. [Tur06]

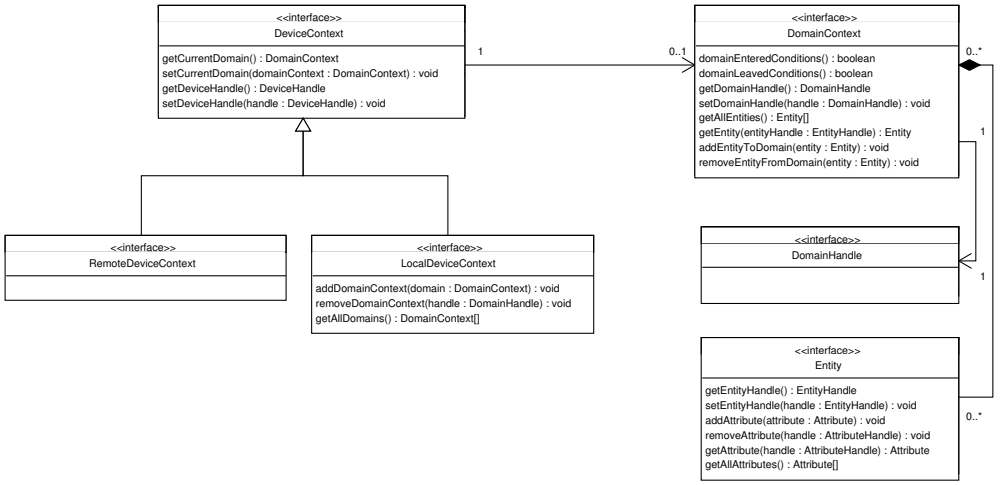


Abbildung 6.6.: Abbildung des Domänenmodells (nach (Tur06))

Abbildung des Domänenmodells Durch die in den lokalen und entfernten Kontexten enthaltenen Domänen (`DomainContext`) wird eine Situation beschrieben, in der ein Ausschnitt aus dem Gesamtkontext relevant ist (vgl. Abbildung 6.6). Sie ist eindeutig durch einen Bezeichner (`DomainHandle`) gekennzeichnet und definiert in jeweils einer Booleschen Funktion die Bedingungen, in denen die Domäne betreten und wieder verlassen wird. Den Domänen können dann Entitäten (`Entity`) hinzugefügt werden, die diesen Kontextausschnitt inhaltlich definieren. Die Entitäten aggregieren wiederum einzelne Attribute des Attributsubmodells und bieten damit den eigentlichen Zugriff auf die elementaren Kontextinformationen. [Tur06]

Abbildung des Attributmodells Die Attribute repräsentieren im generischen Kontextmodell den Zugriff auf einzelne Kontextinformationen und vereinen zum einen den eigentlichen Kontextwert (`AttributeValue`) und zum anderen Angaben zur Qualität der bereitgestellten Kontextinformation (`QualityConstraints`) sowie Metadaten, die zur Verwaltung der Attribute wichtig sind. Dabei finden sich auch die drei Attributformen des konzeptionellen Modells aus Abschnitt 5.6 wieder. Dies sind erstens

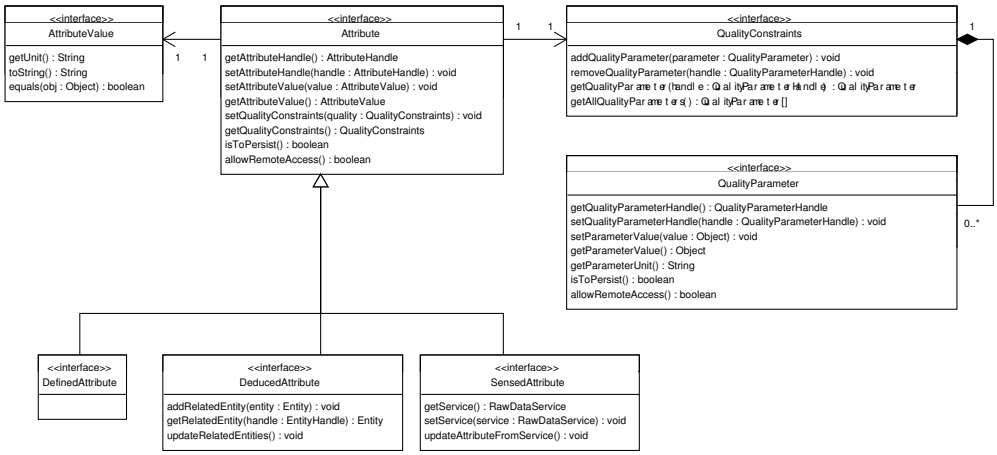


Abbildung 6.7.: Abbildung des Attributmodells (nach (Tur06))

sensorisch oder über Dienste erfasste Attribute (`SensedAttribute`), denen jeweils ein Rohdatendienst als Datenquelle zugeordnet wird. Zweitens können aus bestehenden Entitäten und ihren Attributen weitere Kontextdaten abgeleitet werden (`DeducedAttribute`). Drittens können statische Werte als `DefinedAttributes` abgebildet werden. Zusätzlich zum Wert des Attributes können über Angaben zu Qualitätsrandbedingungen unterschiedliche Qualitätsparameter (`QualityParameter`) angegeben werden, welche die Güte des bereitgestellten Wertes näher beschreiben. Damit stellen sie wichtige Angaben dar, wenn Kontextdaten innerhalb von Anwendungen und Diensten weiterverarbeitet werden. Dies kann zum Beispiel die Standardabweichung eines ermittelten Aufenthaltsortes sein oder eine Angabe zur Fehlerrate eines Messwerts. Zusätzlich wird durch Metaangaben zum Attribut bestimmt, ob es in der Historie für einen späteren Zugriff persistiert werden soll. Zudem kann festgelegt werden, ob das Attribut entfernten Geräten zugänglich gemacht werden darf und es so Teil des föderierten Gesamtkontextes wird. [Tur06]

6.2.1.2. Verwaltungskomponenten

Um Kontextdaten plattform- und programmiersprachenunabhängig verwalten und austauschen zu können, ist eine Trennung der Managementkomponenten von den Modelldaten essenziell. Die Komponenten müssen dabei so abstrakt gestaltet sein, dass sie möglichst beliebige Kontextinhalte verwalten können. Dies ist besonders in einer Systemplattform zur Ausführung Mobiler Prozesse wichtig, da hier nicht vorab bestimmt werden kann, welche konkreten Kontextinformationen Prozesse, die auf ein

Gerät migrieren, benötigen werden. Dabei ist auch bei der Architektur der Verwaltungskomponenten darauf zu achten, autonome funktionale Einheiten zu bilden, die lediglich eine lose Kopplung aufweisen, um die Plattform dynamisch einerseits an die Bedürfnisse der Prozesse als auch andererseits an die Fähigkeiten des mobilen Gerätes anzupassen. Dabei lassen sich aus den Anforderungen an ein Kontextsystem für Mobile Prozesse (vgl. Abschnitt 5.6.1) direkt unterschiedliche Subdienste zur Verwaltung eines Kontextmodells ableiten (vgl. Abbildung 6.8): Zunächst müssen

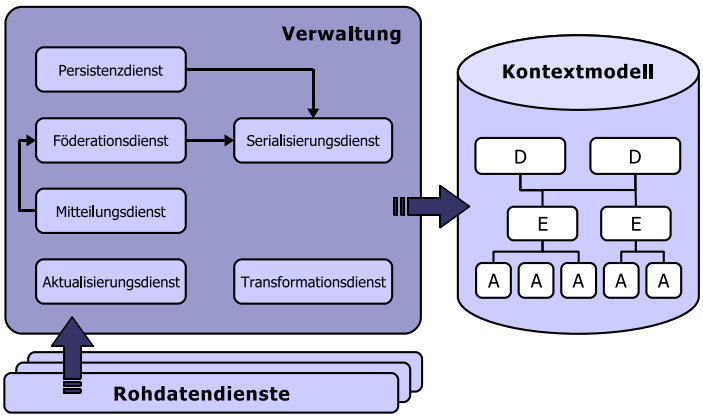


Abbildung 6.8.: Verwaltungskomponenten des Kontext-Managements

Kontextinformationen möglichst aktuell gehalten werden, damit sie auch tatsächlich den Zustand der mobilen Umgebung widerspiegeln. Hierzu müssen je nach Kontextinformation und -quelle individuelle Aktualisierungsintervalle definiert und eingehalten werden. Dazu ist ein *Aktualisierungsdienst* notwendig, der die Daten der Rohdatenquellen in das Kontextmodell überträgt. Zudem müssen Kontextdaten, da sie nicht immer in der Form vorliegen, in der sie von einer Anwendung oder einem Dienst benötigt werden, in äquivalente Repräsentationsformen überführt werden. Deshalb ist die Integration eines *Transformationsdiensts* in die Verwaltungskomponente besonders wichtig. Damit Kontextdatennutzer schnell und flexibel auf Kontextänderungen reagieren können, ist eine Komponente zur proaktiven Anzeige von Zustandsänderungen im Kontextmodell ebenfalls vorzusehen. Dies wird in der Architektur der vorgestellten Plattform durch das Einbeziehen eines *Mitteilungsdiensts* in die Verwaltungskomponente realisiert. Damit die Forderung nach einem Austausch der Kontextdaten zwischen den Geräten einer mobilen Umgebung realisiert werden kann und so aus den zunächst isolierten lokalen Kontextinformationen ein Abbild des Gesamtsystems entsteht, muss ein *Föderationsdienst* integriert werden, der die Kommunikation zwischen den verteilten Kontextsystemen

steuert und so die Grundlage für den Austausch der Kontextinformationen darstellt. Zudem können historische Daten für einzelne Anwendungen wichtig sein, die aus diesen zum Beispiel Vorhersagen für zukünftige Kontextereignisse ableiten. Deshalb müssen Kontextinformationen bei Bedarf persistent gemacht und wieder zugegriffen werden können. Dazu ist ein *Persistenzdienst* notwendig, der ähnlich wie der Aktualisierungsdienst individuell für das einzelne Kontextdatum eine Historie erstellen kann. Dabei müssen sowohl der Persistenzdienst als auch der Föderationsdienst die im Kontextmodell vorhandenen Informationen serialisieren, um sie abspeichern beziehungsweise übermitteln zu können. Hieraus ergibt sich die Notwendigkeit eines *Serialisierungsdiensts*, der Kontextdaten so aufbereiten kann, dass sie an den Persistenzmechanismus eines Systems oder an einen entfernten Kommunikationspartner übertragen und wieder eingelesen werden können.

Aktualisierungsdienst Kontexte sind aufgrund der Mobilität der Systemteilnehmer und der sich natürlich ergebenden Veränderung physikalischer Daten einer mobilen Umgebung einer ständigen Dynamik unterworfen. Damit das auf einem Gerät abgebildete Kontextmodell stets ein möglichst reales Abbild der tatsächlichen Verhältnisse widerspiegelt, muss ein solches Modell also regelmäßig aktualisiert werden. Dabei ist je nach Änderungsrate für jeden Typ eines Kontextdatums ein individuelles Aktualisierungsintervall sinnvoll. So ändert sich zum Beispiel die Umgebungstemperatur nicht ständig, wogegen aber die Position eines mobilen Geräts oftmals einer stetigen Dynamik unterliegt. Deshalb muss es einen Subdienst in einer Kontextverwaltungskomponente geben, der die Informationen im Kontextmodell regelmäßig aktualisiert. Aus diesem Grund bietet die prototypische Implementierung der Systemplattform zur Ausführung Mobiler Prozesse einen Aktualisierungsdienst, der die durch die Rohdatendienste gekapselten und sensorisch oder über Dienste ermittelten Umgebungsdaten zu individuell auf den Typ einer Kontextinformation abgestimmten Zeitpunkten in das Kontextmodell übernimmt. Hierzu wird bei der Definition eines sich verändernden Kontextdatums der Aktualisierungsdienst zusammen mit der gewünschten Aktualisierungsrate an das Datum gebunden. Der Dienst führt dann jeweils nach Ablauf des vorbestimmten Zeitintervalls ein Auffrischen des Kontextwertes durch. Dabei muss nach der Änderung eines Kontextdatums zusätzlich noch durch den Aktualisierungsdienst bestimmt werden, ob und wenn ja welche Domäne zurzeit gültig ist. Dazu werden jeweils die zum Betreten und Verlassen ei-

ner Domäne angegebenen Booleschen Funktionen ausgewertet und je nach Ergebnis die Gültigkeit der entsprechenden Domäne gesetzt. Damit ist der Aktualisierungsdienst also für die Konsistenz des gesamten Kontextmodells verantwortlich und stellt die Verbindung zwischen Rohdaten und den Kontextdaten dar.

Transformationsdienst Ein generisches Kontextmodell muss, da die Repräsentationsform von Kontextdaten nicht vorab durch das Modell selbst bestimmt ist, mit der Polymorphie von Kontextinformationen umgehen können. Dies ist bei der kontextbasierten Kooperation vor allem deshalb notwendig, damit die Ausführung von Mobilen Prozessen oder die Wahl eines Migrationspartners nicht daran scheitert, dass eine vorhandene Information lediglich in einer falschen Repräsentation vorliegt. Deshalb muss eine Kontextverwaltungskomponente die Möglichkeit anbieten, unterschiedliche Repräsentationsformen ein und derselben Information ineinander zu überführen. Dabei ist wichtig, dass nur die Werte einzelner Kontextattribute transformiert werden und nicht ein neues Teilmodell entsteht. Deshalb integriert der Kontextdienst der prototypischen Middleware-Plattform einen Transformationsdienst, der die Werte von Kontextdaten in gleichwertige Darstellungsformen überführt. Die hierzu benötigten Abbildungsfunktionen werden dabei entweder als lokale Funktionen implementiert oder durch das Einbinden von entfernten Diensten realisiert.

Mitteilungsdienst Eines der Hauptziele von kontextbezogenen Systemen ist, auf Kontextänderungen aus Programmen und Diensten heraus reagieren zu können (vgl. Abschnitt 2.5.2 und 3.3.1). Deshalb müssen Änderungen des Kontextes möglichst schnell und flexibel an interessierte Systemkomponenten propagiert werden. Aus diesem Grund ist die Integration einer proaktiven Informationskomponente in den Kontextdienst einer kontextbezogenen Middleware unerlässlich. Die prototypische Implementierung im DEMAC-Projekt enthält deshalb einen Mitteilungsdienst, der Änderungen des Kontextes, die sich durch die Aktualisierung von Kontextdaten ergeben, nach dem *Beobachter-Verhaltensmuster* [GHJV03] an seine Klienten weiterleitet. Dazu registrieren sich die interessierten Komponenten bei der Dienstinstanz und werden durch den Aufruf von entsprechenden Call-Back-Methoden darüber informiert, wenn Domänen betreten beziehungsweise verlassen werden. Durch diese Methode können kontextbezogene Anwendungen und Dienste realisiert werden, ohne die Auswer-

tung der Kontextdaten selbst vornehmen zu müssen. Dies bietet den Vorteil, dass die Entwickler der entsprechenden Komponenten sich auf die anwendungsbezogenen Auswirkungen des Kontextwechsels konzentrieren können und die eigentliche Akquise der Kontextdaten gekapselt bleibt und somit eine hohe Unabhängigkeit und Wiederverwendbarkeit der erweiterten Systemteile erreicht wird.

Föderationsdienst Um der hohe Dynamik in mobilen Systemen gerecht zu werden, sind Architekturen, die aus autonomen und gleichberechtigten Systemen aufgebaut sind, besonders vorteilhaft. Deshalb ist es besonders sinnvoll, wenn Geräte einer mobilen Umgebung jeweils eigenverantwortlich ihre lokalen Kontextdaten verwalten. Um jedoch trotzdem ein Gesamtbild über die Umgebungssituation des lokalen Gerätes zu erhalten, müssen die Systeme ihre Daten im Bedarfsfall untereinander austauschen. Dazu ist die Integration eines Föderationsdienstes für die Übermittlung von Kontextdaten zwischen entfernten Geräten unerlässlich für eine kontextbezogene Architektur. Aus diesem Grund ist in die prototypische Umsetzung der Systemplattform zur Ausführung Mobiler Prozesse ein entsprechender Dienst integriert, der die Daten des Kontextmodells, die dafür freigegeben sind, an andere Systeme übermittelt beziehungsweise entfernte Kontextinformationen in das eigene Modell einbezieht. Hierbei kann das Aktualisierungsintervall der referenzierten Kontextdaten individuell bestimmt werden. Um diese Aktualisierung durchzuführen, implementiert der Föderationsdienst ein Kommunikationsprotokoll zum Anbieten und Beziehen von Kontextdaten auf der Basis des Transport- und Ereignisdiensts der Systemplattform.

Das Kommunikationsprotokoll ist dabei so gestaltet, dass Anfragen nach entfernten Kontextdaten über den Ereignisdienst mittels eines Anfrageereignisses (`RemoteContextRequestEvent`) gestellt werden und die eigentlichen Kontextdaten dann mittels des nachrichtenorientierten Transportdiensts übertragen werden. Dazu registriert sich der Föderationsdienst als `EventListener` für die Anfrageereignisse beim Ereignisdienst, um über eingehende Anfragen proaktiv informiert zu werden. Erhält der Dienst nun mittels des entsprechenden Ereignisses eine Anfrage nach seinem lokalen Kontext, serialisiert er zunächst seinen aktuell gültigen Kontext. Dabei werden jedoch nur diejenigen Attribute berücksichtigt, die für die Föderation zuvor freigegeben wurden. Die so erhaltenen serialisierten Kontextdaten werden dann vom Föderationsdienst als Nutzdaten einer Transportnachricht gekapselt und durch die Angabe eines Bezeichners mit

dem Präfix „`demac:context:remoteContext`“ als Kontextnachricht gekennzeichnet. Diese Nachricht wird dann über den Transportdienst an die Adresse des Anfragenden im Overlay-Netzwerk gesendet. Geht eine solche Nachricht beim Empfänger ein, entnimmt er die Kontextdaten, erzeugt aus ihnen ein `RemoteDeviceContext` und fügt ihn seiner Liste von bekannten entfernten Kontexten hinzu. Bricht die Verbindung zum entfernten Gerät jedoch dauerhaft ab oder ist sie langfristig getrennt, wird der Kontext des entsprechenden Gerätes wieder aus der Liste der bekannten Kontexte gelöscht, sodass ein möglichst aktuelles Abbild der mobilen Umgebung entsteht. [Tur06]

Persistenzdienst Neben aktuellen Kontextdaten können auch historische Kontextdaten für Anwendungen und Dienste interessant sein. So kann zum Beispiel nur durch die Auswertung einer Sequenz von Kontextdaten eine fundierte Aussage über die Bewegung eines Gerätes im Raum gemacht werden. Um jedoch eine solche Kontextdatensequenz zu erhalten, müssen Kontextdaten in diskreten Zeitabständen zwischengespeichert werden. Deshalb ist die Integration eines Dienstes zur persistenten Speicherung von (historischen) Kontextdaten wichtig. Der Kontextdienst besitzt daher der prototypischen Systemplattform einen Persistenzsubdienst, der einzelne Kontextinformationen in individuell vorgegebenen Zeitabständen abspeichert. Hierzu wird ähnlich wie beim Aktualisierungsdienst bei der Definition eines Kontextattributs der Persistenzdienst zusammen mit dem Speicherintervall an das Datum gebunden. Daraufhin sorgt der Persistenzdienst dafür, dass in den vorgegebenen Zeitabständen das Kontextattribut serialisiert und persistiert wird. Der Dienst ist dabei so entworfen, dass die Daten entweder lokal auf dem mobilen Gerät gespeichert werden oder aber ein entfernter Dienst zum Speichern verwendet werden kann. Damit können selbst Geräte ohne eigene Möglichkeit zum Speichern der Daten eine Historie aufbauen, sind dadurch aber von der Verfügbarkeit des entfernten Dienstes abhängig.

Serialisierungsdienst Die Möglichkeit, Kontextstrukturen zu serialisieren, ist eine elementare Aufgabe und unter anderem Grundlage für das persistente Speichern und den Austausch von Modelldaten des lokalen Kontextes. Eine solche Serialisierung überführt dabei das in Form von Softwareobjekten vorliegende Modell in eine äquivalente Repräsentationsform, die übermittelt beziehungsweise auf externe Speicher geschrieben werden kann. Deshalb ergibt sich die Notwendigkeit, einen Subdienst zu

integrieren, der eine solche Abbildung sowohl vom Objektmodell zum serialiserten Modell als auch anders herum bietet. Dementsprechend ist ein Serialisierungsdienst in der prototypischen Implementierung der Middleware zur Realisierung der kontextbasierten Kooperation enthalten. Dabei ist dieser zunächst abstrakt durch entsprechende Schnittstellen definiert und kann durch unterschiedliche Implementierungen verwirklicht werden. Zurzeit ist eine Variante zum Abbilden des Kontextmodells in eine XML-Repräsentation in der Systemplattform vorhanden – es können aber durchaus auch weitere Serialisierungsdienste bereitgestellt werden, die andere Abbildungen realisieren. Um durch das lokale Kontextmodell zu navigieren, verwendet der implementierte Serialisierungsdienst den *Reflection-Mechanismus*, der von der Java-Ausführungsplattform angeboten wird. Hierbei wird durch das Auswerten der in den Kontextkomponenten enthalten Subelemente sukzessive eine vollständige Serialisierung des objektbasierten lokalen Modells erreicht. Hierbei muss jedoch bei der Serialisierung für die Föderation der Kontextdaten darauf geachtet werden, dass nur für die Weitergabe frei gegebene Elemente serialisiert werden, was durch das Auswerten der entsprechenden sondierenden Methode am Modellobjekt geschieht.

6.2.2. Verteilter Dienstzugriff

Die Unterstützung der Ausführung von Aktivitäten Mobiler Prozesse ist die zentrale Aufgabe des Kontextdienstes in der Systemplattform zur Realisierung der kontextbasierten Kooperation. Hierzu wird neben dem föderierten Kontextmodell als zentrale Wissensbasis über nicht-funktionale Kontextdaten auch ein Verzeichnis zum Auflösen von abstrakten Dienstklassen sowie ein generischer Mechanismus zum Einbinden der gefundenen Dienstinstanzen benötigt. Dabei muss solch eine Verzeichniskomponente aufgrund der Heterogenität und Dynamik sowie der geforderten Autonomie der einzelnen Geräte ebenfalls einem verteilten Ansatz folgen, bei dem lokale Informationen auch lokal verwaltet werden und externe Daten nur bei Bedarf dynamisch einbezogen werden. Die Grobarchitektur der Verzeichniskomponente ergibt sich dabei aus dem konzeptionellen Ansatz aus Abschnitt 5.6.3.3. Zugriffspunkt für die Ausführungsumgebung der Mobilen Prozesse ist eine nach dem *Fabrik-Erzeugungsmuster* [GHJV03] konzipierte Komponente, die technologiespezifische Dienstaufrufe durch Objekte mit einer einheitlichen Schnittstelle kapselt (vgl. Abbildung 6.9). Diese Komponente greift dabei auf die *Verzeichnisverwaltung* des verteil-

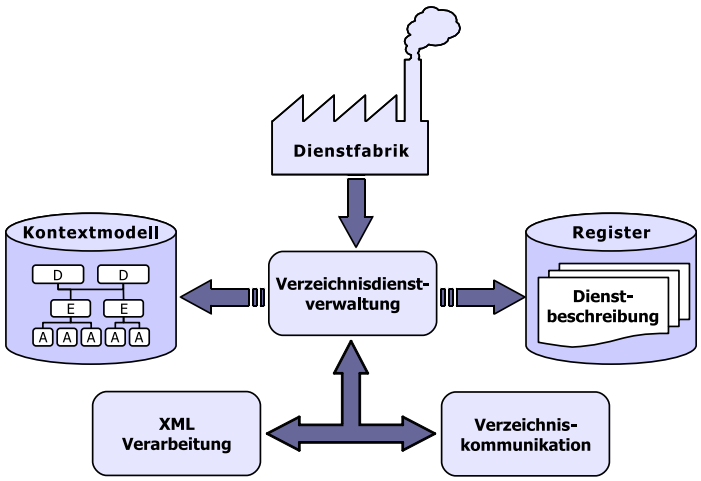


Abbildung 6.9.: Komponenten des Verzeichnisdiensts

ten Verzeichnisdiensts zurück, um zu den Anfragen der Prozessausführung passende Dienstinstanzen zu finden. Diese Managementkomponente greift ihrerseits zunächst auf den lokalen Datenbestand des Systems zu und versucht so in Abwägung der nicht-funktionalen Parameter der Anfrage und der *Dienstbeschreibungen* in der *Registratur* einen passenden Dienst zu finden. Sollte dieser Versuch scheitern, greift der Verwaltungsdienst mit Hilfe einer *Kommunikationskomponente* auf Daten anderer Verzeichnisdienste in der Umgebung zurück und ist so in der Lage, auch entfernte Dienste einzubinden.

Dienstfabrik Aufgabe der Dienstfabrik in der Architektur ist es, zwischen den von der Ausführungsumgebung für Mobile Prozesse benötigten abstrakten und generischen Dienstzugriffen auf die Aufrufe individueller Dienstimplementierungen abzubilden. Hierzu muss die Fabrik, wenn eine passende Dienstinstanz zu einer Anfrage gefunden wurde, ein entsprechendes *Adapter-Objekt* [GHJV03] erzeugen, das dann eine einheitliche Schnittstelle zum Aufruf der Dienstinstanz bietet (`ServiceObject`). Dies ist notwendig, da entsprechend der Fähigkeiten eines Geräts eine unterschiedliche Anzahl von Technologien zum entfernten Dienstaufruf mit ihren jeweiligen Eigenheiten zur Verfügung steht und so nicht vorab bestimmt werden kann, wie die konkreten Aufrufe einer Aktivität tatsächlich aussehen. Deshalb muss die Fabrik so konstruiert sein, dass sie dynamisch möglichst beliebige technologieabhängige Adapterobjekte konstruieren kann. Diese müssen durch instanzspezifische `ServiceDelivery-Messages`, in denen die technologieabhängigen Details zum Dienstaufruf

gekapselt sind, an den konkreten Dienst angepasst werden. An den so entstehenden Objekten mit ihrer generischen Schnittstelle kann die Ausführungsumgebung der Mobilen Prozesse dann einheitlich auf die Dienste, welche die abstrakte Dienstklasse implementieren, zugreifen.

Verzeichnisverwaltung Die Verwaltungskomponente des Verzeichnisdienstes hat die Aufgabe, den Datenbestand über die lokal verfügbaren Dienstinstanzen mit Hilfe einer internen Registratur zu verwalten sowie nicht-funktionale Parameter einer Anfrage mit dem lokalen Kontextmodell abzugleichen. Desweiteren muss sie externe Daten anderer Verzeichnisse bei Bedarf anfordern und auswerten. Hierzu muss die Komponente über ihre Schnittstelle das Auffinden von Dienstinstanzen unter nichtfunktionalen Randbedingungen, das Einstellen lokaler Dienste und das Entfernen einmal registrierter Dienste ermöglichen. Da das Verzeichnis eine zentrale Komponente zur Verwaltung der Dienste des Systems ist, ist es sinnvoll, nur jeweils ein Exemplar des Verzeichnisses pro Gerät bereitzustellen und deshalb diesen Subdienst als *Singleton* [GHJV03] zu konzipieren. Zudem muss die Management-Komponente möglichst flexibel konfigurierbar sein, sodass die Komponentenauswahl auf die Bedürfnisse des Gerätes und der Anwendungen abgestimmt werden kann. So sollte es zum Beispiel möglich sein, das Anbieten von lokalen Dienstinformationen auszuschalten, wenn keine lokalen Dienste vorliegen, um damit die Ressourcen des mobilen Gerätes zu schonen.

Die prototypische Implementierung des DEMAC-Projekts verarbeitet dabei Anfragen von der Fabrik nach Dienstinstanzen folgendermaßen (vgl. [Ada06]): Im ersten Schritt der (lokalen) Anfrageverarbeitung wird eine Suche im internen Register durchgeführt, da die Nutzung lokaler Dienstinstanzen ohne eine Kommunikation über die eher unzuverlässige drahtlose Netzwerkschnittstelle auskommt und so eine erhöhte erfolgreiche Ausführung erwartet werden kann. Hierbei werden zunächst die nicht-funktionalen Rahmenbedingungen der Anfrage mit dem lokalen Kontext abgeglichen. Bei einer positiven Auswertung wird dann in der Registratur nach einer zur abstrakten Dienstklasse passenden Dienstbeschreibung (`ServiceInformation`) gesucht. Sollte diese Suche erfolgreich sein, wird als Ergebnis eine entsprechende Antwortnachricht (`ServiceDelivery-Message`), welche die instanz- und technologiespezifischen Aufrufdetails der gefundenen Dienstinstanz enthält, generiert und an den Anfragenden als Ergebnis übermittelt. Dabei wird zum Erstellen der Antwortnachricht auf die XML-Verarbeitungskomponente zurückgegriffen, die sowohl

das Generieren von XML-Dokumenten aus den Objekten des Verzeichnisses als auch die Gegenoperation (*Parsen*), also das Analysieren und Erzeugen von Objekten aus XML-Dokumenten, zentral bereitstellt. Sollte jedoch durch die interne Suche keine passende Dienstinstanz gefunden werden, wird die Anfrage über die Kommunikationskomponente an andere Verzeichnisse in der Umgebung weitergereicht. Dazu wird eine Anfragenachricht (`ServiceDiscoveryMessage`) als Multicast-Ereignis an die bekannten Geräte versendet. Über ein Beobachterobjekt, das beim Transportdienst registriert wird, wird dann auf das Eintreffen von Antwortnachrichten externer Verzeichnisse gewartet, von denen jedoch nur eine an den Aufrufenden – normalerweise ist dies die Fabrik-Komponente – als Ergebnis der Suche weitergereicht wird. Hierbei kann die Wartezeit, die vergehen soll, bis die Anfrage als gescheitert angesehen wird, an die Bedürfnisse der mobilen Umgebung angepasst werden – so können zum Beispiel bei einer langsameren Netzanbindung längere Wartezeiten sinnvoll sein, als bei einer schnellen Verbindung. Die Verarbeitung von externen Anfragen geschieht dabei so wie im ersten Schritt der lokalen Anfrage, es erfolgt also erst der Abgleich der nicht-funktionalen Parameter mit dem Kontext und dann die Suche einer passenden Dienstinstanz in der Registratur. Dabei wird nur im Erfolgsfall eine Antwortnachricht erzeugt und über den Transportdienst an den Anfragenden weitergeleitet. Hierdurch reduziert sich die Anzahl der versendeten Nachrichten und die Kommunikationsverbindungen werden für relevante Informationen frei gehalten und so effizienter genutzt.

Dienstregister Das Dienstregister ist ein zentraler Bestandteil des Verzeichnisdiensts und muss von der Verwaltungskomponente vor dem direkten Zugriff durch externe Systemkomponenten gekapselt werden, um stets einen konsistenten Datenbestand und damit die korrekte Verarbeitung von Anfragen zu garantieren. Das Register muss dabei die lokal verfügbaren technologie- und instanzspezifischen Dienstinformationen verwalten und den Zugriff über die Dienstklasse in Kombination mit der Implementierungstechnologie bieten, um interne und externe Suchanfragen beantworten zu können. Deshalb muss das Register strukturierte Dienstbeschreibungen (`ServiceInformation`) speichern und wieder entfernen können sowie eine adäquate Suchfunktion bieten. Die Dienstinformationen enthalten hierbei Angaben zur Dienstklasse, Realisierungstechnologie sowie die spezifische Schnittstellen- und Zugriffsbeschreibung. In der prototypischen Implementierung der Systemplattform kapselt dabei das Registerob-

jekt eine dynamische Datenstruktur, die sich der Anzahl der in ihr gespeicherten Dienstinformationen automatisch anpassen kann. Hiermit wird gewährleistet, dass die Architektur die (theoretisch) erreichbare Anzahl von angebotenen Diensten nicht einschränkt, sodass die Menge der verfügbaren Dienste allein von den Fähigkeiten des Gerätes abhängt. [Ada06]

Verzeichniskommunikation Um die gemischte Kommunikation über Ereignisdienst und Transportdienst der Verzeichniskomponente im Fall externer Suchanfragen zu synchronisieren, muss eine Komponente beide Kommunikationsparadigmen zusammenführen. Deshalb nutzt die prototypische Implementierung des Verzeichnisdienstes eine Kommunikationskomponente, die ein Such- und Übermittlungsprotokoll für externe Dienstanfragen implementiert. Dazu wird durch das Registrieren entsprechender Beobachterobjekte und die Verwendung der XML-Verarbeitungskomponente eine objektbasierte Kapselung der Suche erreicht. Dabei müssen die Suchanfragen die Dienstklasse und die nicht-funktionalen Rahmenbedingungen enthalten, die auf der Seite des Empfängers der Anfrage in entsprechende Suchaufrufe an der Management-Komponente umgewandelt werden. Die Ergebnisse der Anfragen müssen dann auf entsprechende Antwortnachrichten an den Suchenden abgebildet und diese dann übertragen werden. Dabei wird auf der Seite des Anfragenden das Absenden der Suchanfrage und der Erhalt der Antwort synchronisiert, sodass die externe Suche in der Management-Komponente auf einen einfachen Methodenaufruf reduziert wird. [Ada06]

6.2.3. Fazit zum Kontextdienst der Systemplattform

Der Kontextdienst ist ein wichtiger Teil der Systemplattform, da er nichtfunktionale Informationen über die (mobile) Umgebung liefert und das Auflösen von abstrakten Diensten in konkrete Dienstinstanzen bereitstellt. Durch das generische Kontextmodell, mit dem beliebige Kontextdaten abgebildet und strukturiert zugegriffen werden können, unterstützt er die Ausführung und Migration Mobiler Prozesse besonders gut. Dies ergibt sich dabei vor allem aus der Tatsache, dass Mobile Prozesse beliebige Aktivitäten und nicht-funktionale Rahmenbedingungen aufweisen können sowie nicht-deterministisch im Gesamtsystem verteilt ausgeführt werden. Damit ist eine Vorhersage der benötigten Kontextdaten im Allgemeinen nicht möglich, sodass die generische und abstrakte Struktur des Kontextmodells eine sehr gute Grundlage für die Umsetzung der kontext-

basierten Kooperation bietet. Hierzu zählt auch der konsequente Einsatz der Föderation von lokal verwalteten Kontext- und Dienstinformationen, sodass autonome Geräte entstehen, die aber im Bedarfsfall auf die Informationen der Umgebung zugreifen können. Somit werden die Dynamik und Heterogenität der einzelnen Geräte im Gesamtsystem besonders berücksichtigt, sodass eine Architektur entsteht, die den intrinsischen Eigenschaften mobiler Systeme möglichst optimal Rechnung trägt. Dieser Ansatz wird dabei durch die Verwendung von lose gekoppelten und somit parametrisierbaren Komponenten auch konsequent in der Systemarchitektur fortgesetzt. Damit entsteht insgesamt ein Kontextsystem, das nicht nur die Kategorien klassischer kontextbezogener Anwendungen – wie die kontextbezogene Selektion, Präsentation, Aktion und Annotation (vgl. Abschnitt 3.3.1) – nicht nur unterstützt, sondern auch die neue Kategorie der kontextbasierten Kooperation ermöglicht.

6.3. Der Prozessdienst

Um Mobile Prozesse in eine Systemplattform zur Umsetzung der kontextbasierten Kooperation zu integrieren, muss die Architektur eine Komponente vorsehen, die Prozessbeschreibungen interpretieren und ausführen kann. Zudem muss diese dafür Sorge tragen, dass die Migrationseigenschaft der Mobilen Prozesse realisiert wird. Deshalb ist in der Architektur der prototypischen Systemplattform ein *Prozessdienst* integriert, der diese Dienstleistung auf der Basis der anderen Systemkomponenten umsetzt (vgl. Abbildung 6.1). Dabei muss zum einen das Metamodell zur *Beschreibung Mobiler Prozesse* (vgl. Abschnitt 5.5) umgesetzt werden und zum anderen muss eine möglichst *modular aufgebaute Ausführungskomponente* bereitgestellt werden, die den erweiterten Prozesslebenszyklus (vgl. Abschnitt 5.4.4) umsetzt. Auch bei diesen Plattformkomponenten ist dabei darauf zu achten, dass sie möglichst so konzipiert sind, dass sie den intrinsischen Eigenschaften mobiler Systeme gerecht werden. Insbesondere die unterschiedliche Leistungsfähigkeit der beteiligten Komponenten und die hohe Dynamik der Umgebung müssen hierbei beachtet werden. Durch die Integration eines solchen Prozessdiensts wird die Architektur der Systemplattform komplettiert, sodass durch das Zusammenwirken aller betrachteten Plattformkomponenten eine angemessene Grundlage für die Entwicklung von Anwendungen mit kontextbasierter Kooperation entsteht.

6.3.1. Abbildung des Metamodells zur Beschreibung Mobiler Prozesse

Eine besonders wichtige Voraussetzung für die kooperative und somit verteilte Ausführung lang andauernder Prozesse über heterogene (mobile) Systeme ist die Umsetzung des Metamodells Mobiler Prozesse (vgl. Abschnitt 5.5) in eine Beschreibungssprache, mit der sich reale Prozesse definieren lassen. Eine solche Beschreibungssprache muss aufgrund der unterschiedlichen und vielfältigen Geräte einer mobilen Umgebung abstrakt und damit geräteunabhängig sein, um eine Migration der Prozesse zwischen heterogenen Systemen und Plattformen zu ermöglichen. Da das Metamodell Mobiler Prozesse sich bereits an dem entsprechenden Modell der Prozessaustauschsprache XPDL (vgl. Abschnitt 4.5.3) orientiert, ist es sinnvoll, deren XML-basierte Umsetzung als Basis für die Realisierung der Beschreibungssprache Mobiler Prozesse zu verwenden. Aus diesem Grund ist die *DEMAC Process Description Language* (DPDL), welche die Beschreibungssprache der prototypischen Umsetzung der kontextbasierten Kooperation darstellt, als entsprechendes XML-Schema definiert (vgl. Abbildung 6.10). Dabei muss die Sprache die von XPDL geerbten Sprachelemente so erweitern, dass mit ihr einzelne verteilt ausführbare Prozessinstanzen beschrieben werden können. [ZK07, KZL07b, KZL06]

Bei der Erweiterung der Basissprache ist darauf zu achten, dass die Eigenschaften und Anforderungen, die sich aus der Heterogenität der Systeme, der mobilen Umgebung und der durch Migration realisierten verteilten Ausführungsstrategie ergeben, berücksichtigt werden. Dabei ist vor allem die Autonomie der an der Ausführung der Mobilen Prozesse beteiligten Systeme zu bewahren. Deshalb muss es zu jedem Zeitpunkt möglich sein, den Ausführungszustand des Prozesses eindeutig zu bestimmen und mit dem Prozess zu anderen Systemen zu migrieren. Aus diesem Grund ist der Ausführungszustand Teil der Prozessbeschreibung. Dabei ist der Zustand des Gesamtprozesses durch den Zustand seiner Aktivitäten des Kontrollfusses (`ActivityRefs`) und die während der Ausführung benötigten Daten (`DataFields`) definiert. Deshalb muss in der Prozessbeschreibungssprache jeder Aktivität ihr jeweiliger Zustand zugeordnet sein. Dieser ist in der prototypischen Umsetzung durch einen vereinfachten Aktivitätslebenszyklus definiert (vgl. Abbildung 6.11). Um nach der Migration eines Prozesses möglichst schnell die Ausführung des Mobilen Prozesses wieder aufzunehmen und die Ressourcen der Geräte zu schonen, muss eine aufwendige Traversierung der Aktivitäten zur Bestimmung der als nächstes auszuführenden Aufgabe vermieden werden. Dies wird erreicht,

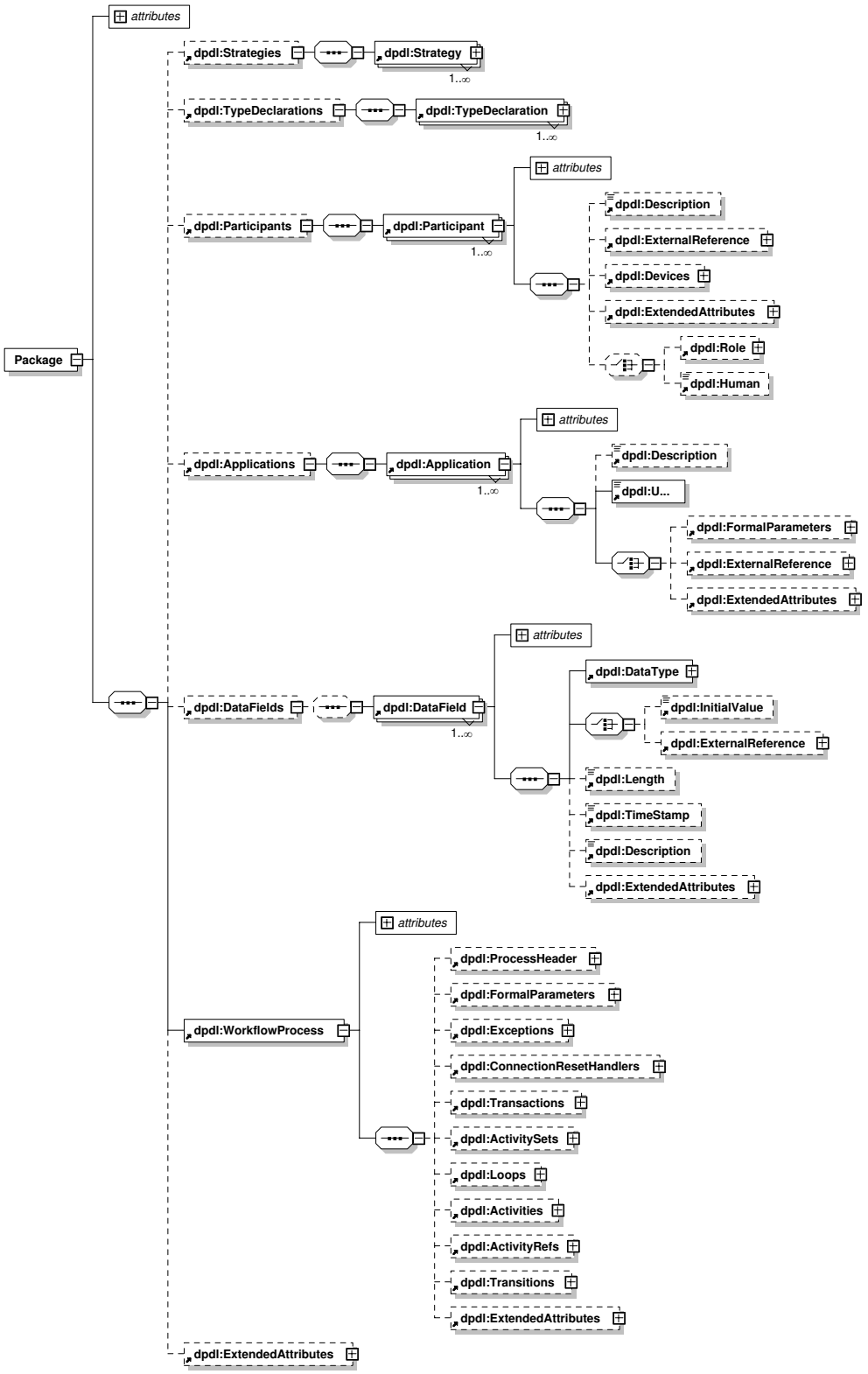


Abbildung 6.10.: XML-Schemaauszug zur Beschreibung Mobiler Prozesse

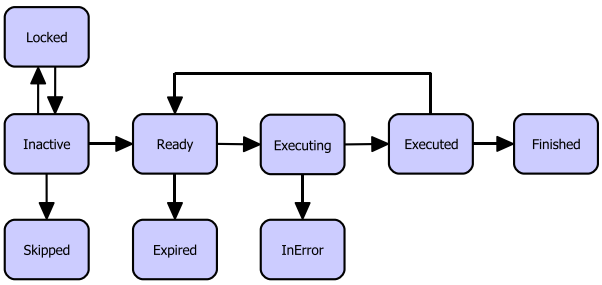


Abbildung 6.11.: Vereinfachter Aktivitätslebenszyklus (nach (ZK07, Hol07))

indem der jeweilige Startpunkt innerhalb des Kontrollflusses in der Prozessbeschreibung explizit durch die entsprechende Aktivitätsreferenz angegeben wird. [ZK07, KZL07b, KZL06]

Desweiteren muss die Beschreibungssprache möglichst schlank gehalten werden, damit sie auch von ressourcenschwachen Geräten verarbeitet werden kann. Hierzu muss die Sprache so aufgebaut sein, dass Aktivitäten innerhalb des Kontrollflusses wiederverwendbar sind. Dies wird durch die Trennung der allgemeinen Beschreibung der Aktivität und ihrer Verwendung innerhalb des Kontrollflusses in der prototypischen Umsetzung realisiert. Hierzu werden die Aktivitäten zunächst abstrakt definiert und dann durch Referenzen in den Kontrollfluss integriert. Zudem wird die zunächst graphbasierte zeitlich-logische Abfolge der Aktivitäten durch Blockstrukturen, zum Beispiel zur Definition von Schleifen, ergänzt. Als weiterer Mechanismus zum schonenden Umgang mit den Ressourcen der beteiligten Systeme muss es möglich sein, dort, wo es sinnvoll ist, umfangreiche Daten zu externalisieren. Hierzu ist es möglich, Datenquellen nicht nur über die internen Datenfelder des Prozesses zu definieren, sondern auch als Referenz auf externe Datenquellen. Hierbei muss der Modellierer des Prozesses jedoch abwägen, inwieweit die Externalisierung der Daten die Ausführung durch eine eventuelle Nichtverfügbarkeit dieser Informationen negativ beeinflusst. Jedoch kann es zum Beispiel sinnvoll sein, an explizit angegebenen Synchronisationspunkten, deren dauerhafte Erreichbarkeit gegeben ist, auf externe Datenquellen zurückzugreifen. Um solche Synchronisationspunkte definieren zu können, muss deshalb die Möglichkeit bestehen, Geräte und Teilnehmer gezielt einzelnen Aktivitäten zuzuordnen (`Participants`). Hierbei können diese in DPDL entweder durch eindeutige Referenzen, wie zum Beispiel eine digitale Identität, explizit angegeben werden oder aber abstrakt durch eine Rolle definiert werden, die dann die Menge der möglichen Ausführungsorte der Aktivität nicht-funktional einschränkt. Zudem können nur durch die Ver-

wendung von solchen Synchronisationspunkten parallele Prozesspfade, die keine Datenabhängigkeit aufweisen, unabhängig voneinander ausgeführt werden. Schließlich muss dieser Mechanismus auch deshalb in die Sprache integriert werden, damit Ergebnisse des Mobilen Prozesses an den Prozessinitiator oder einen anderen vorbestimmten Teilnehmer gezielt übermittelt werden können. [ZK07, KZL07b, KZL06]

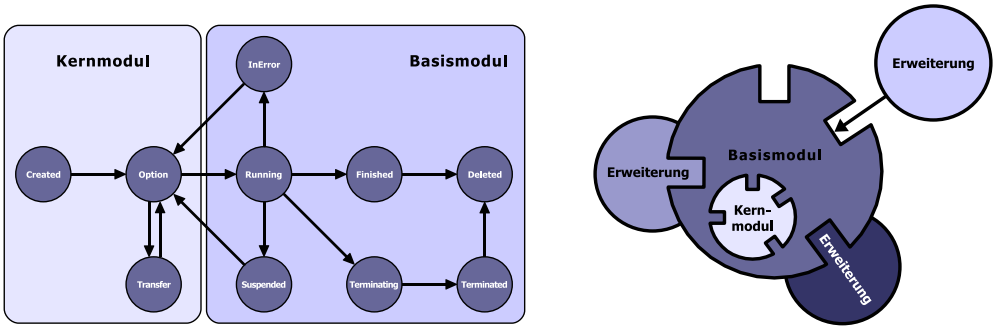
Eine weitere Forderung, die sich aus der Dynamik und den damit verbundenen unzuverlässigen Kommunikationsverbindungen der Umgebung Mobiler Prozesse ergibt, ist, dass diese Ausnahmesituationen explizit in die Modellierung der Prozessinstanz aufgenommen werden müssen. Hierbei sind zwei unterschiedliche Basissituationen zu unterscheiden: Erstens können Anwendungsfehler bei der Ausführung einzelner Aktivitäten auftreten und zweitens können sich Plattform- beziehungsweise Verbindungsfehler ereignen. Da diese unterschiedlichen Fehlerarten meist auch eine unterschiedliche Behandlung erfordern, müssen jeweils eigene Behandlungsroutinen in die Prozessbeschreibung aufgenommen werden. Diese Fehlerbehandlung wird in der Beschreibungssprache der prototypischen Systemplattform durch die Definition von Subprozessen realisiert, sodass prinzipiell beliebig auf die Fehler eingegangen werden kann und so der ursprüngliche Prozessverlauf nach der Fehlerbehandlung wieder aufgenommen werden kann. [ZK07, KZL07b, KZL06]

Um einen Einfluss auf die Prozessausführung und Migration innerhalb der heterogenen Umgebung der Mobilen Prozesse nehmen zu können, müssen nicht-funktionale Parameter in die Prozessbeschreibung integriert werden. Diese werden in DPDL als Strategien (`Strategies`) zunächst allgemein definiert und dann entweder dem globalen Prozess oder einzelnen Aktivitäten zugeordnet. Hierdurch wird wiederum die Wiederverwendbarkeit garantiert und zugleich eine schlankere Beschreibung erreicht. Globale Strategien werden dabei vor allem bei der Auswahl von Migrationspartnern eingesetzt, wogegen Aktivitäten zugeordneter Strategien meist die Menge der potenziellen Dienste zur Ausführung der jeweiligen Aktivität nicht-funktional einschränken. [ZK07, KZL07b, KZL06]

6.3.2. Umsetzung der Prozessausführung

Neben der Möglichkeit, Mobile Prozesse beschreiben zu können, muss der Prozessdienst einer Systemplattform auchmechanismen für die Interpretation und Ausführung der Prozessbeschreibungen bereitstellen. Hierzu muss ein Workflow-Management-System integriert werden, das die Pro-

zesse auf dem lokalen Gerät verwaltet und ihre verteilte Ausführung koordiniert. Dabei muss entsprechend der opportunistischen Verarbeitungsstrategie (vgl. Abschnitt 5.4.1) zunächst versucht werden, die Aktivitäten des Prozesses in Zusammenarbeit mit dem Kontextdienst auf dem lokalen Gerät auszuführen und in den Fällen, wo dies nicht möglich ist, einen geeigneten Migrationspartner zu finden. Um hierbei die Potenziale einer möglichst großen Anzahl von Systemen einer (mobilen) Umgebung nutzen zu können, müssen die Komponenten des Prozessdienstes aufgrund der Heterogenität und der unterschiedlichen Leistungsfähigkeiten der Geräte modular und lose gekoppelt sein. Hierbei bietet der erweiterte Lebenszyklus Mobiler Prozesse mit seinen neuen Zuständen eine inhärente Möglichkeit, das Workflow-Management zu strukturieren (vgl. Abbildung 6.12(a)): Die Verwaltungs- und Ausführungskomponente wird in zwei grundlegende Module aufgeteilt, wobei in einem *Kernmodul* die Migration der Prozesse gekapselt wird. In einem ergänzenden *Basismodul* wird dann die eigentliche Ausführung der Aktivitäten des Mobilen Prozesses bereitgestellt. Um diese Grundarchitektur auf leistungsfähigen Systemen mit Zusatzfunktionalitäten erweitern zu können, müssen über einen entsprechenden Plug-in-Mechanismus zusätzliche *Erweiterungsmodule* in die Architektur integrierbar sein (vgl. Abbildung 6.12(b)). So entsteht eine flexible Dienstarchitektur, die minimal nur die Migration der Prozesse unterstützen muss und trotzdem als Zugangspunkt zu den Potenzialen der mobilen Umgebung und als Vermittler zwischen anderen (eventuell leistungsfähigeren) Geräten dienen kann. Potentere Systeme hingegen können durch die Erweiterungsmöglichkeit individuell abgestimmte Zusatzfunktionalitäten anbieten, die das Gesamtsystem bereichern. [ZK07, KZL07b, KZL06]



(a) Abbildung des Prozesslebenszyklus

(b) Architektur der Ausführungskomponente

Abbildung 6.12.: Ausführung von Mobilen Prozessen (z.T. nach (ZK07) und (KZL06))

Kernmodul Das Kernmodul der Prozessdienstarchitektur muss alle Funktionalitäten enthalten, damit das lokale System sich in minimaler Form an der kontextbasierten Kooperation beteiligen kann (vgl. Abbildung 6.13). Damit dient dieses Modul als zentraler Zugangspunkt zu den Diensten und Potenzialen der gesamten (mobilen) Umgebung, weshalb diese Komponente in der prototypischen Umsetzung ebenfalls als *Singleton* [GHJV03] konzipiert ist. Zudem bietet die Implementierung eine ein-

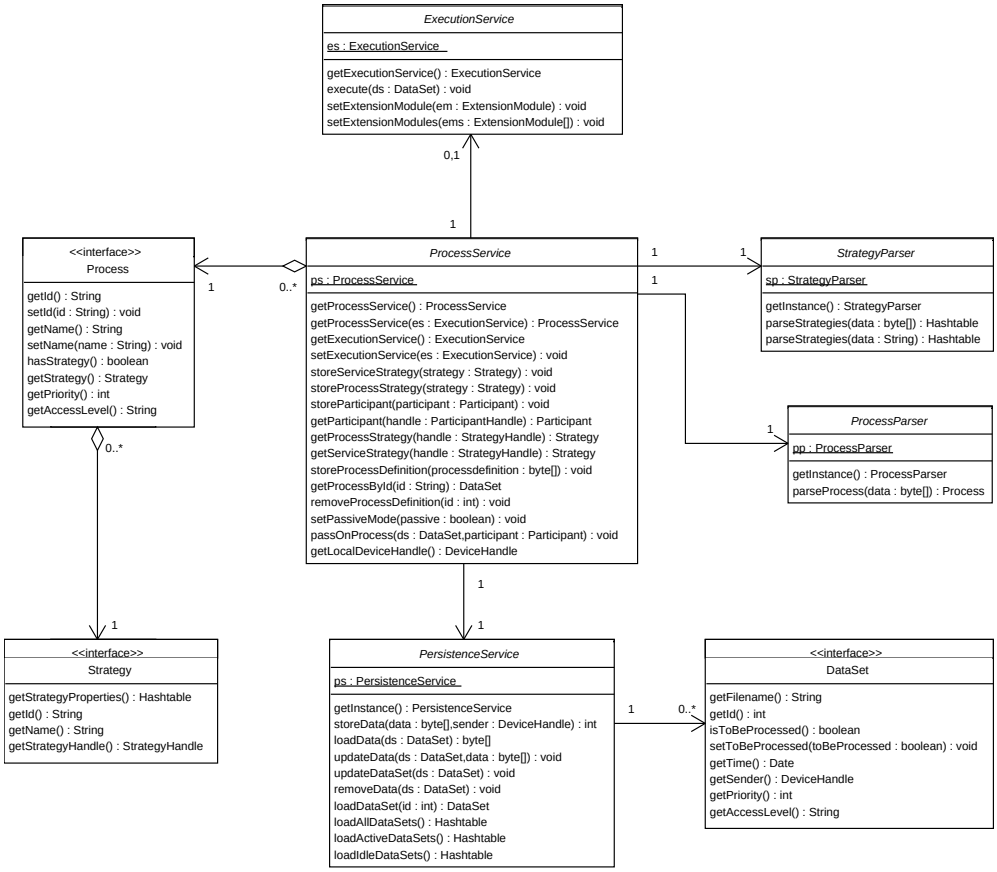


Abbildung 6.13.: Klassendiagramm des Kernmoduls

heitliche Schnittstelle, mit deren Hilfe Beschreibungen Mobiler Prozesse in ihrer XML-Repräsentation an die Middleware übergeben werden können (`storeProcessDefinition()`) sowie alle vom Prozessdienst momentan bearbeiteten Prozesse und ihre Metainformationen verwaltet werden können. Dabei muss das Kernmodul, unter anderem aufgrund der begrenzten Energiereserven mobiler Geräte, besonders auf die Persistenz der ihm anvertrauten Prozesse achten, damit in Ausnahmesituationen keine Prozesse verloren gehen. Deshalb muss das Kernmodul das Speichern der Prozesse auf nicht-flüchtigen Medien unterstützen. Dies wird

in der prototypischen Umsetzung des Kernmoduls durch einen Persistenzsubdienst (`PersistenceService`) realisiert, der das Speichern und Laden von Prozessen und ihren Metadaten kapselt und auf die jeweilige Ausführungsplattform angepasst werden kann. [ZK07]

Neben der lokalen Verwaltung persistenter Prozessdaten muss die Kernkomponente, um die Mobilen Prozesse zur Ausführung zu bringen, regelmäßig überprüfen, ob sie entweder lokal ausgeführt werden können oder ob es andere Geräte gibt, zu dem der Prozess zur weiteren Bearbeitung migriert werden kann. Dabei kann eine lokale Ausführung nur dann eingeleitet werden, wenn neben dem Kernmodul auch das Basismodul als eigentliche Prozessausführungskomponente verfügbar ist. Um diese Ausführbarkeit zu prüfen, muss vom Kernmodul zunächst die Erfüllbarkeit der nichtfunktionalen Parameter des Mobilen Prozesses mit Hilfe des Kontextdienstes ausgewertet werden. Deshalb müssen die Prozessbeschreibungen nicht nur persistent gespeichert werden, sondern die Prozessmetainformationen und die globalen nicht-funktionalen Parameter müssen zusätzlich intern als Objekte bereitstehen, was in der prototypischen Implementierung durch den `ProcessParser` und den `StrategyParser` ermöglicht wird. Sind die nicht-funktionalen Parameter lokal erfüllt, muss, sollte ein Basismodul vorhanden sein, der Prozess an dieses zur weiteren Verarbeitung übergeben werden, da in der Ausführungsstrategie eine lokale Verarbeitung der Migration vorgezogen wird. Sind die nicht-funktionalen Parameter jedoch nicht erfüllt, muss ein Migrationsversuch eingeleitet werden, damit der Prozess eventuell auf einem anderen Gerät vorangebracht und damit die Ausführungswahrscheinlichkeit des Mobilen Prozesses erhöht werden kann (vgl. Abschnitt 5.4.2). [ZK07]

Im Zuge der Migration muss die Kernkomponente mehrere Aufgaben ausführen: Dies sind auf der Seite des Senders die Suche nach passenden Migrationspartnern und der eigentliche Transfer der Prozessdaten sowie auf Seiten des Empfängers das Entgegennehmen und Verarbeiten der ankommenden Prozessbeschreibungen. Dabei müssen passende Migrationspartner mit Hilfe des Kontextdienstes bestimmt werden, da die Menge der potenziellen Partner durch nicht-funktionale Parameter eingeschränkt sein kann und diese nur über den Kontext evaluiert werden können. Die eigentliche Suche wird dabei in der prototypischen Implementierung durch ein Multicast-Ereignis (`RequestParticipantEvent`) eingeleitet, das die geforderten nicht-funktionalen Anforderungen an den Migrationspartner in der mobilen Umgebung verteilt. Diese Anfragen werden auf den entfernten Geräten lokal ausgewertet. Führt die Auswertung zu einem positiven

Ergebnis, meldet sich das Gerät beim Anfragenden mit Hilfe eines Antwortereignisses (`ReceiveParticipantEvent`). Der eigentliche Transfer der Prozessdaten findet daraufhin mit Hilfe des Transportdiensts und eines einfachen Migrationsprotokolls (vgl. Abbildung 6.14) statt. Das Protokoll bietet dabei eine Grundsicherung der Migration, durch die eine gesicherte Übertragung der Beschreibungen erreicht wird. Das Ziel des Protokolls ist dabei, die Prozessbeschreibung erst dann lokal zu entfernen, wenn sie beim Empfänger persistent gespeichert ist, damit der Prozess nicht während des Transfers verloren gehen kann. Zudem wird – durch die explizite Anzeige von Fehlern beim Löschen der Prozessdaten auf der Seite des Absendens der Daten – die Wahrscheinlichkeit stark reduziert, dass der Prozess durch die Migration verdoppelt wird. [ZK07]

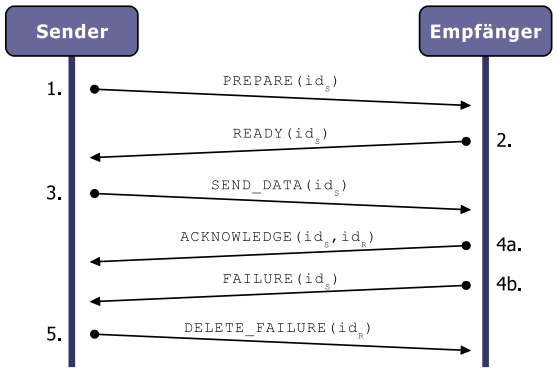


Abbildung 6.14.: Einfaches Migrationsprotokoll (nach (ZK07))

Basismodul Das Basismodul baut auf dem Kernmodul auf und erweitert dieses zu einer vollwertigen Workflow Engine (vgl. Abbildung 6.15), indem es die fehlenden Zustände des erweiterten Prozesslebenszyklus (vgl. Abbildung 6.12(a)) realisiert. Es muss dementsprechend den Mobilen Prozess im Ganzen interpretieren und ausführen können. Deshalb muss diese Komponente die gesamte Beschreibung auf ihre Integrität hin prüfen und in eine entsprechende Objektstruktur überführen. Dazu verwendet die prototypische Implementierung einen passenden XML-Parser (`DpdlParser`), um auch die von der Kernkomponente nicht benötigten Informationen aus der Beschreibung zu extrahieren. So stehen dem Basismodul sämtliche Informationen zum Mobilen Prozess über das `WorkflowProcess`-Objekt zur Verfügung. [ZK07]

Eine der grundlegenden Aufgaben des Basismoduls ist die fachliche und technische Entscheidung darüber, ob die anstehenden Aktivitäten lokal ausgeführt werden können oder ob der Mobile Prozess zu anderen Geräten

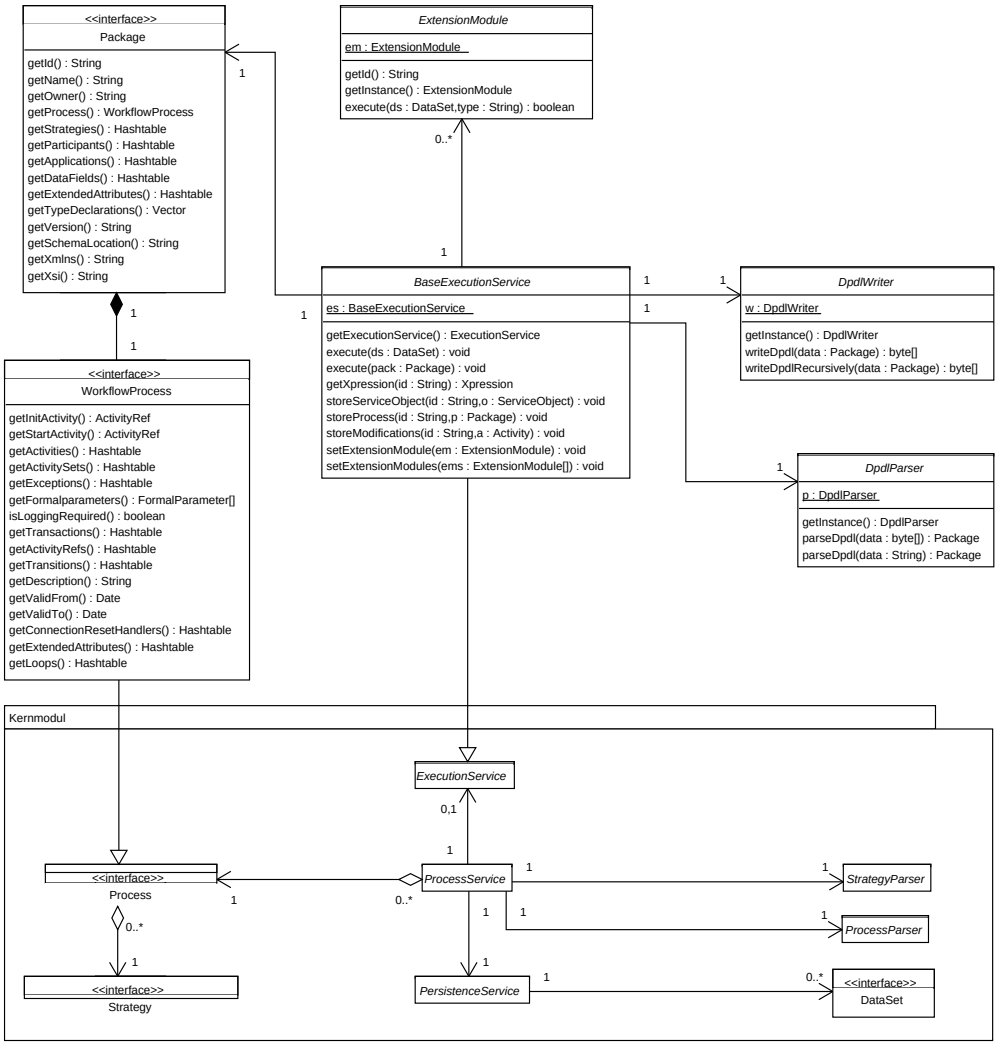


Abbildung 6.15.: Klassendiagramm des Basismoduls

migriert werden muss. Dazu muss zu Beginn der Ausführung zunächst entschieden werden, ob der Prozess noch gültig ist, also ob die aktuelle Zeit zwischen dem Start- und Endzeitpunkt des erlaubten Ausführungszeitraums liegt. Erst danach kann die Interpretation der Aktivitäten und die Navigation durch den Kontrollfluss beginnen. Dabei wird mit der in der Beschreibung ausgezeichneten Startaktivität angefangen und danach die weiteren Aktivitäten anhand der Transitionen und ihrer Übergangsbedingungen bestimmt. Hierbei muss die Basiskomponente dafür sorgen, dass alle Änderungen am Zustand des Gesamtprozesses – also Änderungen der Prozessdaten und der Status der Aktivitäten – stets persistent gehalten werden, damit keine Fortschrittsinformationen in Ausnahme- oder Fehlersituationen verloren gehen. Hierdurch wird also vermieden, dass

Aktivitäten mehrfach ausgeführt werden und so zum Beispiel entstehende Kosten wiederholt anfallen. Die Persistenz des Zustands wird in der prototypischen Implementierung des Basismoduls durch die bereits aus dem Kernmodul bekannten `DataSets` und dem `PersistenceService` realisiert. Da der Fortschritt in der Ausführung eines Mobilen Prozesses vor seiner Migration wieder in eine entsprechende XML-basierte Beschreibung überführt werden muss, muss der Basisdienst – im Gegensatz zur Kernkomponente, die keine Veränderungen an der Beschreibung vornimmt – aus den lokalen Objekt- und Datenstrukturen wieder eine textuelle Beschreibung erzeugen können. Dementsprechend besitzt die prototypische Implementierung des Basismoduls eine Serialisierungskomponente (`DpdlWriter`), die XML-Beschreibungen aus den lokalen Informationen erzeugen kann. [ZK07]

Der eigentliche Fortschritt in der Ausführung Mobiler Prozesse geschieht durch die Ausführung der Aktivitäten des Kontrollflusses. Um Aktivitäten abzuarbeiten, muss das Basismodul mit unterschiedlichen Aktivitätstypen (der Blockaktivität und der atomaren Aktivität) umgehen können. Hierbei kapseln Blockaktivitäten lediglich wiederverwendbare Abschnitte aus dem Kontrollfluss, sodass die Ausführung der Blockaktivität durch das Ausführen ihrer internen Startaktivität geschieht. Atomare Aktivitäten hingegen werden entweder nur zur Steuerung des Kontrollflusses (`RouteActivities`) eingesetzt und bedürfen daher quasi keiner Ausführung oder sie repräsentieren einen Dienst, der eine Teilaufgabe des Prozesses erbringt. Um dabei Dienste einzubinden, die eine atomare Aktivität implementieren, muss das Basismodul zunächst die zur Aktivität gehörende abstrakte Dienstklasse auflösen (vgl. Abschnitt 5.5.4). Hierzu muss es zunächst alle notwendigen Informationen aus der Prozessbeschreibung extrahieren. Dies sind unter anderem die Daten der Dienstklasse, der gewünschte Ausführungsort sowie die nicht-funktionalen Parameter. Anhand dieser Informationen muss dann zunächst bestimmt werden, ob eine lokale Ausführung erlaubt ist – also der lokale `Participant` mit dem geforderten Prozessteilnehmer übereinstimmt. Ist diese Entscheidung positiv ausgefallen, muss das Basismodul am Kontextdienst die Suche nach einer Implementierung des Dienstes initiieren und erhält gegebenenfalls ein generisches Adapterobjekt (vgl. Abschnitt 6.2), an dem es den eigentlichen Dienstaufruf vornimmt. Darf das lokale Gerät die Aktivität nicht ausführen beziehungsweise kann kein geeigneter Dienst vom Kontextdienst bereitgestellt werden, muss der Basisdienst die Migration des Mobilen Prozesses initiieren. Dies geschieht in der prototypischen Im-

plementierung dadurch, dass der Prozess zusammen mit seinem Zustand serialisiert und zurück an das Kernmodul übergeben wird. Das Kernmodul versucht dann, den Mobilen Prozess an ein anderes Gerät zu migrieren, oder übergibt den Prozess zur lokalen Ausführung erneut an das Basismodul, falls sich der Ausführungskontext des lokalen Gerätes geändert hat und so eine Ausführung der Aktivität wieder möglich wird. Dabei trifft die Kernkomponente keine fachliche beziehungsweise technische Entscheidung über die Möglichkeit der Ausführung, diese ist vielmehr dem Basismodul vorbehalten. [ZK07]

Um diese Grundfunktionalität des Basis- und Kernmoduls auf leistungsstarken beziehungsweise spezialisierten Systemen durch abgegrenzte Zusatzfunktionen erweitern zu können, muss das Basismodul die Möglichkeit bieten, Erweiterungsmodule einzubinden. Diese Module werden dabei in der prototypischen Umsetzung über eine standardisierte Schnittstelle (`ExtensionModule`) integriert und angesprochen (vgl. Abbildung 6.15). Hierdurch ist es dann zum Beispiel möglich, die Workflow-Komponente der vorgestellten Architektur mit einem Modul zur Bereitstellung transaktionalen Verhaltens zu erweitern. [ZK07]

6.3.3. Integration transaktionalen Verhaltens

Das Zusammenfassen von Aktivitäten zu Transaktionen und ihre gekoppelte Ausführung sind wichtige Erweiterungen elementarer Mobiler Prozesse (vgl. Abschnitt 5.7). Deshalb ist auch eine entsprechende Erweiterung der elementaren Ausführungsumgebung Mobiler Prozesse um eine Transaktionsunterstützung besonders wichtig. Da transaktionales Verhalten jedoch nicht von allen Mobilen Prozessen benötigt wird und somit diese Funktionalität eine Zusatzkomponente darstellt, kann diese architektonisch sinnvoll als Erweiterungsmodul an das Basismodul angebunden werden. Entsprechend ist der Transaktionssubdienst (`TransactionSupportService`) als Erweiterungskomponente in der prototypischen Middleware-Plattform zur Umsetzung der kontextbasierten Kooperation implementiert. Hierbei muss die Transaktionsverwaltung *Sequenz- und Join-Transaktionen mit diskreter und globaler Kompensation* bereitstellen, um einen adäquaten Funktionsumfang zu bieten (vgl. Abschnitt 5.7.3). Insgesamt muss dabei die Ausführung der Aktivitäten des Mobilen Prozesses klar von der Transaktionsverwaltung getrennt werden und es müssen lediglich die für eine Transaktion relevanten Informationen ausgetauscht werden. Diese Informationen, wie die Grenzen der Transaktion, Start und

Ende der Ausführung einer Aktivität sowie Fehler, die während der Verarbeitung auftreten, werden in der prototypischen Implementierung über entsprechende Ereignisse zwischen Basismodul und Transaktionsverwaltung ausgetauscht. Mit diesen Informationen muss die Transaktionskomponente den Status einer Transaktion überwachen und gegebenenfalls kompensierende Maßnahmen einleiten. Um eine diskrete beziehungsweise globale Kompensation mit wiederholtem Ausführen einer Transaktion zu ermöglichen, muss das Transaktionsmodul die relevanten Prozessdaten sowie weitere Metainformationen vor Beginn der Transaktion speichern. Hierbei müssen diese Daten, vor allem dann, wenn der Prozess während einer Transaktion migriert werden soll, Teil der Prozessbeschreibung werden. Dabei muss es möglich sein, Prozesspfade, die vor dem Abschluss der Transaktion diese verlassen, zeitlich in ihrer Ausführung zu verzögern (*Backout-Schutz*). Nur so kann verhindert werden, dass Aktivitäten auf nicht endgültigen Daten arbeiten. Hierzu muss der Aktivitätslebenszyklus mit einem Sperrzustand versehen werden (*Locked*), der nur durch den erfolgreichen Abschluss der Transaktion wieder verlassen werden kann (vgl. Abbildung 6.11). [Hol07]

Die im Rahmen der prototypischen Umsetzung der Systemplattform implementierte Transaktionssubkomponente realisiert zurzeit die Integration von Sequenztransaktionen mit Forward Recovery und globaler Kompensation. Dabei wird vom Basismodul während der Navigation durch den Kontrollfluss des Mobilen Prozesses und der Ausführung der Aktivitäten die Kontrolle an das Transaktionsverwaltungsmodul zeitweise abgeben. Dies ist notwendig, damit zum Beispiel die Prozessdaten für das Forward Recovery (vgl. Abschnitt 5.7.3) gespeichert werden oder kompensierende Maßnahmen durchgeführt werden können. Kommt es zur Kompensation einzelner Aktivitäten beziehungsweise der ganzen Transaktion, weist das Transaktionsmodul das Basismodul an, mit Hilfe des Kontextdienstes die kompensierenden Dienste zu finden und auszuführen. Somit wird die Transaktionsunterstützung realisiert, ohne die grundsätzlichen Verantwortlichkeiten der Subdienste des Prozessdienstes zu verändern. [Hol07]

Werden Mobile Prozesse in (relativ) zuverlässigen Umgebungen, wie zum Beispiel infrastrukturbasierten Netzwerken, eingesetzt, können auch bestehende verteilte Transaktionsmodelle und -paradigmen eingebunden werden. Diese Einschränkung ist deshalb zu machen, da in dynamischen Umgebungen die Gefahr besteht, dass der Prozess in einen blockierenden Zustand übergeht. Sollen trotzdem traditionelle verteilte Transaktionen integriert werden, muss lediglich über ein entsprechendes Erweite-

rungsmodul der Transaktionsverwalter der verteilten Transaktion angesprochen werden, der die Steuerung dann zum Beispiel mittels des Zwei-Phasen-Commit-Protokolls (2PC) realisieren kann. Dieser Transaktionsverwalter kann dabei direkt im lokalen Transaktionsmodul implementiert sein oder aber das lokale Erweiterungsmodul dient allein als Schnittstelle, welche die Kommunikation mit dem Verwalter umsetzt. Die Anbindung an die Prozessausführung kann dabei entsprechend der zuvor beschriebenen prototypischen Realisierung der erweiterten und speziell für Mobile Prozesse entwickelten Konzepte erfolgen: Auch hier muss zunächst beim Betreten der Transaktion die Kontrolle vom Basismodul an das Erweiterungsmodul übergeben werden. Nach dem Eintreffen der Information über den Ausgang der Transaktion wird bei einem Zurücksetzen der Transaktion die Kompensation ebenfalls wieder mittels des Prozess- und Kontextdiensts angestoßen beziehungsweise im Erfolgsfall die Transaktion im Prozessdienst als erfolgreich ausgeführt markiert und alle gesperrten Aktivitäten wieder freigegeben, sodass der Mobile Prozess weiter ausgeführt werden kann.

6.3.4. Fazit zum Prozessdienst der Systemplattform

Der Prozessdienst ist als Ausführungskomponente Mobiler Prozesse das zentrale Architekturelement zur Realisierung kontextbasierter und kooperativer Anwendungen. Durch die konsequente funktionale Gliederung in Kern-, Basis- und Erweiterungsmodule bietet er eine große Flexibilität, um auch leistungsschwache Geräte in das Gesamtsystem einzubinden und somit einer hohen Anzahl von Teilnehmern einen Zugang zu den Potenzialen der (mobilen) Umgebung zu ermöglichen. Diese Vielseitigkeit und Anpassbarkeit wird durch die Trennung der Migrationsunterstützung von der eigentlichen Ausführung der Prozesse erreicht und durch die klare Struktur des erweiterten Prozesslebenszyklus begünstigt. Dabei bleibt das Konzept der Mobilen Prozesse aufgrund ihrer Umsetzung als abstrakte XML-basierte Beschreibung und die Verwendung der von konkreten Technologien entkoppelten Kommunikationsparadigmen konzeptionell unabhängig von der konkreten prototypischen Implementierung. Vielmehr ist es möglich, weitere Implementierungen zu realisieren, um auch bisher nicht einbindbare Systeme zu integrieren und so weitere Dienste und Ressourcen mobiler Umgebungen zu nutzen.

Insgesamt bietet der Prozessdienst damit eine gute Basis, um die Heterogenität und Dynamik mobiler Systeme bezüglich der Definition und

Ausführung Mobiler Prozesse transparent zu halten. Zudem werden durch die Komponentenbildung und vielseitige Erweiterbarkeit der Architektur sowie durch den Einsatz von Kontextwissen während der Ausführung und Migration die Forderungen des Mobile Computing nach dynamischer Anpassbarkeit (*Adaptability*) und technologischer Unabhängigkeit des Systems besonders vorteilhaft unterstützt.

7. Evaluierung des Ansatzes zur kontextbasierten Kooperation

Um bewerten zu können, ob und inwieweit das Konzept und die Realisierung Mobiler Prozesse eine kontextbasierte Kooperation ermöglichen, wird in diesem Kapitel der Ansatz verschiedenartig evaluiert. Hierbei wird zunächst aufgezeigt, in welchem Umfang die im konzeptionellen Kapitel (vgl. Kapitel 5) aufgestellten Anforderungen an migrierende und verteilt ausgeführte Prozesse im Mobile Computing von den eingeführten Konzepten und Komponenten erfüllt werden. Des Weiteren werden die aus der stochastischen Analyse abgeleiteten Eigenschaften mobiler Prozesse mit Hilfe des im Projekt DEMAC (*Distributed Environment for Mobility-Aware Computing*) entwickelten Prototyps experimentell evaluiert. Abschließend wird ergründet, inwieweit die Umsetzung klassischer kontextbezogener Anwendungen in mobilen Systemen durch die bereitgestellten Kommunikations- sowie Kontextkomponenten unterstützt werden sowie komplexe Aufgaben in Form von Mobilen Prozessen durch das Gesamtsystem abgebildet und ausgeführt werden. Hierbei wird unter anderem auf Erfahrungen aus unterschiedlichen Projekten in Forschung und Lehre des Arbeitsbereichs *Verteilte Systeme und Informationssysteme* des *Departments Informatik* der *Universität Hamburg* zurückgegriffen.

7.1. Bewertung des Konzepts der kontextbasierten Kooperation und ihrer Realisierung als Mobile Prozesse

Der Erfüllungsgrad der eingeführten Konzepte und Realisierungsmöglichkeiten bezüglich der in Kapitel 5 aufgestellten Anforderungen an das Konzept der kontextbasierten Kooperation und der entsprechenden Umsetzung als Mobile Prozesse wird anhand eines konsolidierten Anforderungskatalogs (vgl. Tabelle 7.1), der gleichartige Anforderungen aus den konzeptionell herausgearbeiteten Teilbereichen zusammenfasst, gruppiert und bewertet. Hierbei wird zunächst auf die Anforderungen eingegangen, die sich hauptsächlich auf Mobile Prozesse beziehen. Danach werden die erweiterten Anforderungen an die Bereitstellung und Nutzung von Kontextwissen betrachtet, bevor sich abschließend mit den Anforderungen befasst

wird, die sich aus der mobilen Umgebung im Allgemeinen herleiten lassen.

Eine besonders wichtige und elementare Anforderung an die kontextbasierte Kooperation und ihre Umsetzung als Mobile Prozesse ist diejenige nach einer Unterstützung komplexer und damit strukturierter Aufgaben unter Ausnutzen möglichst aller Potenziale der (mobilen) Umgebung (E 1, E 2, E 3). Dieser Aspekt wird durch die Prozessorientierung und den Ansatz der verteilten Ausführung erfüllt. Die aus dieser verteilten Ausführung abgeleiteten Forderungen nach einer Unterstützung der Migration (E 9) sind durch die Mobilen Prozesse ebenfalls erfüllt. Die zur Realisierung der Mobilen Prozesse geforderte abstrakte Beschreibung der Aufgaben (E 4) mit Hilfe von Prozessbasisstrukturen (E 6) sowie nichtfunktionalen Rahmenbedingungen (E 10) wird auch in vollem Umfang geboten. Das durch die hohe Abstraktionsebene und die Migration gebotene späte Binden (E 8) von Dienstklassen ist zumindest für nicht-manuelle Aktivitäten erfüllt. Benutzerinteraktionen beziehungsweise manuelle Aufgaben werden zwar grundsätzlich unterstützt, jedoch fehlt es noch einer Möglichkeit, diese Interaktion direkt auch im Mobilen Prozess auszudrücken, sodass die Anforderung E 8.a momentan nur teilweise erfüllt ist. Dies hat natürlich auch auf den Erfüllungsgrad der Forderung nach einer grundsätzlichen Entkopplung von Funktionalität und Darstellung (E 5) Einfluss, sodass hier eine volle Unterstützung im Moment noch nicht gegeben ist.

Die Umsetzung des Konzepts der kontextbasierten Kooperation durch Mobile Prozesse stellt natürlich die Frage nach einer Unterstützung von Transaktionen (E 7), um Aktivitäten zu größeren und zusammenhängenden Ausführungseinheiten zusammenzuschließen. Insgesamt ist hier durch die Einführung von Sequenz- und Join-Transaktion die Abbildung einer Vielzahl von transaktionalen Verhaltensmustern möglich, jedoch werden allgemeine und flexible Transaktionsbereiche (E 7.f) hiermit nicht abgedeckt. Die bereitgestellten Transaktionsmechanismen bieten jedoch die volle Unterstützung langlebiger Transaktionen (E 7.d), unterschiedlicher Dienstarten (E 7.e) und der Migrationseigenschaft (E 7.c). Zudem unterstützen sie die für langlebige Transaktionen besonders wichtigen Eigenschaften der Atomarität (E 7.a) und Konsistenz (E 7.b) in vollem Umfang.

Die Ausführung mobiler Prozesse ist stark mit der Bereitstellung von generischen Kontextinformationen (E 11) verbunden und wird durch das eingeführte Kontextmodell sowie die entsprechende Middleware-Komponente voll unterstützt. Die hierbei geforderte Erweiterbarkeit und Wiederver-

wendbarkeit (E 12) wird auch durch die Abstraktion von konkreten Informationstypen (E 16) zusammen mit der Möglichkeit der Transformation von Kontextdaten (E 17) realisiert. Dabei wird die Flexibilität und Skalierbarkeit durch die Trennung von Kontextmodell und -management (E 13) sowie die Föderation der Kontextdaten (E 15) und das Anbieten von Filtern zum Abbilden von relevanten Kontextausschnitten (E 19) umgesetzt. Auch die geforderte Möglichkeit zur Abbildung unterschiedlicher Kontextabstraktionen (E 20) und der Einbeziehung von Qualitätsangaben (E 21) ist im Modell der vorgestellten Systemplattform enthalten. Die ebenfalls geforderte und unterstützte Integration eines verteilten Verzeichnisses (E 22) zur Verwaltung und zum Auflösen von abstrakten Dienstklassen (E 22.b) ist für die Ausführung Mobiler Prozesse besonders wichtig, kann aber die Integrationsfähigkeit des allgemeinen Kontextkonzepts in bestehende Systeme (E 14) leicht einschränken, sodass hier leichte Abstriche im Erfüllungsgrad zu machen sind. Die geforderte technologieneutrale Bereitstellung von Dienstbeschreibungen (E 22.c) unter Einbezug der Kontextdaten (E 22.d) ist hingegen wieder gegeben. Auch der schonende Umgang mit den Ressourcen der Systeme, vor allem in Bezug auf die Kommunikation (E 22.a), ist durch die verteilte Ausführungsstrategie gewährleistet, da hier die relevanten Daten zwischen den Kommunikationspartnern ausgetauscht werden.

Die Umsetzung des Konzepts der kontextbasierten Kooperation in einer mobilen Umgebung mit ihren intrinsischen Eigenschaften stellt ebenfalls Anforderungen an die Konzepte und deren Umsetzungen, die insgesamt auch adäquat berücksichtigt werden (E 23). Dabei ist durchgängig auf eine Technologieneutralität der Konzepte und Komponenten (E 23.b) sowie die Unabhängigkeit von Netzwerkstrukturen (E 23.f) geachtet worden, sodass diese Forderungen in vollem Umfang unterstützt werden. Durch einfache Protokolle und die Unterstützung von proaktiven Kommunikationsformen (E 23.g) wird zudem der von der Middleware erzeugte Kommunikationsaufwand minimiert (E 23.e). Aufgrund der prototypischen Entwicklung sind die robusten und fehlertoleranten Konzepte (E 23.d und E 23.c) in der Umsetzung noch verbesserungsfähig, sodass hier bei der Bewertung noch leichte Abstriche gemacht werden müssen. Bei der Umsetzung von Sicherheits- und Vertrauensmechanismen sind beim derzeitigen Prototypen größere Abstriche zu machen, sodass diese nur als teilweise umgesetzt angesehen werden können.

Insgesamt kann also festgehalten werden, dass das Konzept der kontextbasierten Kooperation und seine Umsetzung als Mobile Prozesse in wei-

ten Teilen die identifizierten Anforderungen an ein erweitertes System zur Unterstützung von Nutzern in mobilen Umgebungen bereits weitgehend erfüllt.

Kürzel	Referenz	Titel	Bewertung
		Mobile Prozesse	
E 1	MW 1, MM 1	Unterstützung komplexer Aufgaben	++
E 2	KS 2	Strukturierung in Teilaufgaben	++
E 3	MW 3	Nutzen möglichst aller Potenziale der Umgebung	++
E 4	MW 2, KA 1	Abstrakte Beschreibung von Aufgaben	++
E 5	KS 1	Entkopplung von Funktionalität und Darstellung	+
E 6	MM 2	Prozessbasisstrukturen	++
E 7	MM 10	Transaktionen	+
E 7.a	TA 1	Gewährleisten der Atomarität	++
E 7.b	TA 2	Sicherstellen der Konsistenz	++
E 7.c	TA 3	Unterstützung der Migrationseigenschaft	++
E 7.d	TA 4	Unterstützung langlebiger Transaktionen	++
E 7.e	TA 5	Unterstützung unterschiedlicher Dienstarten	++
E 7.f	TA 6	Unterstützung flexibler Transaktionsbereiche	−
E 7.g	TA 7	Toleranz gegenüber Knoten-ausfällen	+
E 8	KA 3, KA 4, MM 5, KS 3	Spätes Binden	++
E 8.a	MM 8	Benutzerinteraktion / manuelle Aufgaben	○
E 9	MM 3	Migration	++

++ = voll unterstützt + = unterstützt ○ = teilweise unterstützt − = nicht unterstützt

Fortsetzung von Tabelle 7.1 auf der nächsten Seite ...

Kürzel	Referenz	Titel	Bewertung
E 10	MM 4	Nicht-funktionale Parameter	++

E 11	KM 1	**Bereitstellen von Kontextdaten**	++
E 12	KM 2	Erweiterbarkeit und Wiederverwendbarkeit	++
E 13	KM 3	Trennung von Modell und Management	++
E 14	KM 4	Integrationsfähigkeit in bestehende Systeme	+
E 15	KM 5, VV 2	Föderation verteilter Kontextdaten	++
E 16	KM 6	Abstraktion von konkreten Informationstypen	++
E 17	KM 7	Transformation von Kontextdaten	++
E 18	KM 10	Flexibilität und Skalierbarkeit	++
E 19	KM 11	Filter zum Abbilden von Kontextausschnitten	++
E 20	KM 12	Abbilden verschiedener Kontextabstraktionen	++
E 21	KM 14	Einbeziehen der Qualität von Kontextdaten	++
E 22	VV 1	Verteiltes Verzeichnis	++
E 22.a	VV 3	Minimierung der Kommunikation	++
E 22.b	VV 4	Unterstützung abstrakter Dienstklassen	++
E 22.c	VV 5	Technologieneutrale Bereitstellung von Dienstbeschreibungen	++
E 22.d	VV 6	Einbinden des Kontexts	++

++ = voll unterstützt + = unterstützt ◯ = teilweise unterstützt − = nicht unterstützt

Fortsetzung von Tabelle 7.1 auf der nächsten Seite ...

Kürzel	Referenz	Titel	Bewertung
E 23	MW 4, MM 11	**Berücksichtigen der Eigenschaften mobiler Systeme**	+
E 23.a	MM 7	Sicherheit / Vertrauen	◯
E 23.b	MM 6	Technologieunabhängigkeit	++
E 23.c	MM 9	Fehlerbehandlung	+
E 23.d	KM 9	Robustheit und Kompaktheit	+
E 23.e	TA 8, VV 2	Minimierung der Kommunikation	++
E 23.f	TA 9	Unabhängigkeit von Netzwerkstrukturen	++
E 23.g	KM 8	Proaktive Kommunikation	++

++ = voll unterstützt + = unterstützt ◯ = teilweise unterstützt − = nicht unterstützt

Tabelle 7.1.: Bewertung des Ansatzes kontextbasierter Kooperation

7.2. Untersuchung zur Steigerung der Ausführungswahrscheinlichkeit Mobiler Prozesse durch Migration

Eine der zentralen Annahmen des Konzeptes kontextbasierter Kooperation ist, dass die Ausführungswahrscheinlichkeit von Mobilen Prozessen durch die Migration steigt (vgl. Abschnitt 5.4.2). Die hierfür zugrunde gelegte erste rein stochastische Analyse soll an dieser Stelle auch noch anhand der realisierten DEMAC-Systemplattform experimentell überprüft werden.

7.2.1. Aufbau der Evaluierungsumgebung

Zur Untersuchung der Ausführungswahrscheinlichkeiten einer Aktivität eines Mobilen Prozesses wurde eine Evaluierungsumgebung eingesetzt, die aus einem Netzwerk von Geräten mit der DEMAC-Plattform besteht. Um auf diesem Gesamtsystem das Migrations- und Ausführungsverhalten bezüglich einer einzelnen Aktivität betrachten zu können, wurden Prozesse mit nur einer Aktivität eingesetzt. Hierbei wurde bewusst auf die Angabe von nicht-funktionalen Randbedingungen verzichtet, um die Migration und damit die Verteilung der Migrationswahrscheinlichkeit nicht zu

beeinflussen. Insgesamt sah die Evaluierungsumgebung dabei im Detail folgendermaßen aus:

- Es wurde ein Netzwerk von sechs Geräten mit der DEMAC-Systemplattform eingesetzt, in dem die Ausführungswahrscheinlichkeit insgesamt 40% betrug. Dabei ergaben sich die 40% dadurch, dass bei der Migration Quell- und Zielgerät nicht identisch sein dürfen und somit fünf Migrationsmöglichkeiten von jedem Teilnehmer aus bestanden.
  - Im Netzwerk waren drei Linux-Systeme, welche die jeweilige Aktivität der zur Evaluierung verwendeten Prozesse nicht ausführen konnten.
  - Zwei weitere Linux-Systeme waren hingegen in der Lage, die jeweilige Aktivität der zur Evaluierung verwendeten Prozesse auszuführen und dienten somit als Senken des Systems.
  - Ein Windows-System diente einerseits als Zugangspunkt zum Gesamtsystem und andererseits als weiteres System, dass die Aktivität der Evaluierungsprozesse nicht ausführen konnte.
- Ein Evaluierungslauf bestand aus 100 Prozessen, die jeweils selbst nur aus einer Aktivität bestanden.
  - Die Prozesse wurden zwar zentral durch nur ein System, das die Aktivität des Evaluierungsprozesses nicht ausführen konnte, in das Netzwerk eingebracht, dieser Umstand wurde jedoch bei der Auswertung rechnerisch korrigiert.
  - Die Migrationswahrscheinlichkeit, die in einem mobilen System zum Beispiel durch unzuverlässige Kommunikationsverbindungen gegeben ist, wurde durch das Verwerfen der Ausführung von 12% der Prozesse abgebildet.
- Die Ausführung der Aktivitäten geschah jeweils lokal, sodass Seiteneffekte durch Fehler bei einer Kommunikation mit einem entfernten Dienst nicht auftreten konnten.

7.2.2. Ergebnisse und Auswertung der Untersuchung

Mit der Evaluierung der Ausführungswahrscheinlichkeit anhand der prototypischen Implementierung ist das Ziel verbunden, zu klären, ob die durch die stochastische Analyse zu erwartende erhöhte Grenzausführungswahrscheinlichkeit für die Ausführung einzelner Aktivitäten tatsächlich auch im realen System eintritt.

Dazu wurde die Anzahl der Migrationsschritte (*Hops*) protokolliert, die der einzelne Prozess und mit ihm die einzige in ihm enthaltene Aktivität bis zur Ausführung benötigt hat. Dieses Verfahren wurde für drei Durchläufe mit je 100 Prozessen durchgeführt und der gemittelte prozentuale Anteil für jede Anzahl an Hops ermittelt. Durch das Bilden der aufsummierten prozentualen Werte bis zum jeweiligen Migrationsschritt ergibt sich dann die entsprechende Grenzausführungswahrscheinlichkeit. Diese wurden schließlich mit der durch die stochastische Analyse erwartete Grenzausführungswahrscheinlichkeit verglichen (siehe Abbildung 7.1).

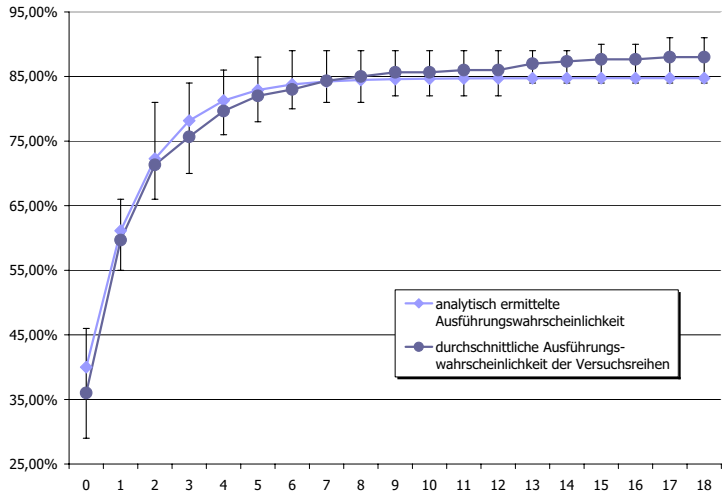


Abbildung 7.1.: Ergebnisse der Untersuchungen zur Ausführungswahrscheinlichkeit bei einer durchschnittlichen Migrationswahrscheinlichkeit von 88% und einer Ausführungswahrscheinlichkeit von 40%. (vgl. Tabelle A.2)

Die Ergebnisse der Evaluierung zeigen, dass die aus der analytischen Untersuchung begründete Annahme einer erhöhten Grenzausführungswahrscheinlichkeit der Aktivitäten durch die durchgeführten Experimente voll bestätigt wird. Es ist zwar zwischen den Läufen eine Schwankungsbreite der Grenzwahrscheinlichkeit der Hops zwischen 5% und 17% aufgetreten, jedoch hat diese keinen Einfluss auf die zu bestätigende grundlegende Aussage, dass durch die Migration eine allgemein höhere Ausführungswahrscheinlichkeit der einzelnen Aktivitäten und somit auch des gesamten Mobilen Prozesses besteht. Zudem wird auch die Aussage, dass schon wenige Migrationsschritte ausreichen, um eine deutlich erhöhte Ausführungswahrscheinlichkeit zu erreichen, durch die Experimente bestätigt. Damit hat also die Umsetzung der kontextbasierten Kooperation durch mobile Prozesse die an sie gestellten Erwartungen in diesen Punkten voll erfüllt.

7.3. Untersuchung der entwickelten Systemplattform

Um die DEMAC-Systemplattform auf ihre Eignung im Hinblick auf ihren praktischen Einsatz in mobilen Umgebungen zu evaluieren, ist sie in unterschiedlichen Entwicklungsstadien anhand von Szenarien und Beispielapplikationen auf ihre funktionale Einsatzfähigkeit, Zuverlässigkeit und Skalierbarkeit hin untersucht worden. Dabei wurden jeweils die für eine Entwicklungsstufe spezifischen Fragestellungen untersucht, um zu ergründen, ob die jeweilige Stufe ausreichende Dienstleistungen bereitstellt, um als Basis für die nächste höhere Ausbaustufe zu dienen. Dabei wurde zunächst die Kommunikationsplattform zum Aufbau eines Overlay-Netzwerks in mobilen Umgebungen (vgl. Abschnitt 6.1) untersucht, dann die um das generische Kontextmodell erweiterte Ausbaustufe (vgl. Abschnitt 6.2) und abschließend die Gesamtplattform zur Ausführung mobiler Prozesse und damit zur Realisierung der kontextbasierten Kooperation (vgl. Abschnitt 6.3).

7.3.1. Zuverlässigkeit und Skalierbarkeit der Kommunikationsplattform

Die Untersuchung der Kommunikationsplattform, bestehend aus dem *asynchronen Transportdienst* und dem *Ereignisdienst* (vgl. Abbildung 6.1), wurden im Rahmen des Lehreprojekts *Realisierung verteilter mobiler Systeme* des Arbeitsbereichs *Verteilte Systeme und Informationssysteme* der *Universität Hamburg* durchgeführt. Dabei wurde auf der Basis der Komponenten zur Etablierung eines nachrichten- und ereignisorientierten Overlay-Netzwerks der *DEMAC-Middleware* ein dezentrales Instant-Messenger-System entwickelt und implementiert. Das Projekt war hierbei so konzipiert, dass es ein reales Abbild mobiler Umgebungen verwirklicht. Dies wurde durch den Aufbau einer hybriden Systemarchitektur erreicht, in der autonome mobile Umgebungen – bei Verfügbarkeit – über Backbone-Infrastruktursysteme zu einem erweiterten Gesamtsystem gekoppelt werden können (vgl. Abbildung 7.2). Aus Sicht der Untersuchung der grundlegenden Systemplattform bestand dabei das Ziel des Projekts darin zu untersuchen, ob sich der Aufbau und die Verwaltung des Overlay-Netzwerks sowie der damit verbundene Nachrichtentransport über unterschiedliche Netzwerke hinweg dynamisch einer sich stetig verändernden Umgebung anpassen kann und mit einer steigenden Anzahl von Teilnehmern skaliert. Damit adressiert diese Untersuchung die elementaren Anforderungen nach Transparenz und Offenheit, Fehlertoleranz und Skalierbarkeit sowie asynchroner Kommunikation und die Unterstützung der Mobilität,

die grundlegend an eine Middleware im Mobile Computing gestellt werden muss (vgl. Tabelle 3.1).

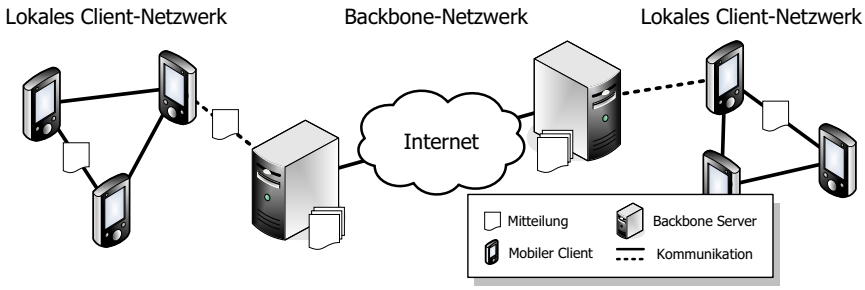


Abbildung 7.2.: Aufbau des Instant-Messenger-Netzwerks zur funktionalen Evaluierung der Kommunikationsplattform.

Insgesamt wurde durch die Entwicklung in eigenständigen Kleingruppen eine heterogene Anwendungslandschaft geschaffen, die lediglich über zuvor definierte Protokolle auf Basis der Kommunikationsplattform gekoppelt sind. Dabei wurde zwischen Gruppen zur Umsetzung der Komponenten zum Austausch von Mitteilungen zwischen mobilen Endgeräten (*Client-Netzwerk*) und Gruppen zum Etablieren eines Infrastruktursystems zur Bereitstellung erweiterter Dienste (*Backbone-Netzwerk*) unterschieden, die folgende Kernaufgaben in zwei Entwicklungsstufen umgesetzt haben:

- Durch das *Anlegen neuer Benutzer* werden dezentral neue Teilnehmer erzeugt und in das System eingebunden.
- *Statusinformationen* geben Auskunft über verschiedene Verfügbarkeits- beziehungsweise Sichtbarkeitsstufen der Teilnehmer.
- Die Integration einer *Benutzerverwaltung* ermöglicht den Aufbau und die Verarbeitung von individuellen Teilnehmerlisten (*Buddy-Liste*).
  - Zur Verwaltung der Liste können Teilnehmer zur Liste *hinzugefügt* und *gelöscht* werden.
  - Zudem ist es möglich, die Kommunikation mit einzelnen Teilnehmern zu *sperren*.
- Zwischen den Nutzern des Client-Netzwerkes ist ein direkter *Austausch von Textnachrichten* und das Bilden von *Konferenzen* möglich.
- Neben dem Austausch von Textnachrichten ist es zudem möglich, einen *Dateitransfer* durchzuführen.

- Ist ein Server des Backbone-Netzwerks verfügbar, können zusätzlich auch Nachrichten an zurzeit nicht verfügbare Teilnehmer gesendet werden (*Offline-Nachrichten*), die dann im Infrastrukturnetz repliziert werden und so später von beliebigen Infrastrukturservern wieder abgerufen werden können.
- Zusätzlich können auch lokale Buddy-Listen im Backbone-Netzwerk abgelegt und wieder bezogen werden (*serverseitige Buddy-Listen*).

Durch die inhärent dynamische Struktur eines Instant-Messenger-Systems, die durch das Eintreten und Verlassen von Teilnehmern an unterschiedlichen Orten, mit unterschiedlichen Geräten und zu beliebigen Zeitpunkten geprägt ist, ist ein solches System besonders geeignet, die gewünschten Eigenschaften zu untersuchen.

Projektergebnisse Die Umsetzung des Instant-Messenger-Systems hat gezeigt, dass die Systemplattform insgesamt den Anforderungen einer dezentralen und verteilten Anwendung mit unterschiedlichen Geräten und Netzwerkzugängen voll gerecht wird. Die Transparenz des Systems zeigte sich dabei durch die nahtlose Integration unterschiedlicher Geräte, wie mobile Klienten, Desktop-PCs und Server. Dabei zeigte das System seine Offenheit und Fehlertoleranz, indem erreichbare Geräte dynamisch in das System eingebunden werden und es regulär und in Fehlerfällen auch irregulär wieder verlassen können, ohne die Gesamtfunktionalität zu beeinträchtigen. Die angebotene Mischung von asynchronen Kommunikationsformen mit einerseits einem nachrichtenorientierten Transportsystem und andererseits einem Ereignisdienst hat sich insgesamt als vorteilhaft erwiesen. So ist der Versand von Textmitteilungen und Dateien zum Beispiel mittels der asynchronen Nachrichten realisiert, wogegen Statusänderungen auf Ereignissen basieren. Dies bedeutet, dass für jede Aufgabe passende Kommunikationsformen zu Verfügung stehen und der Entwickler nicht auf ein Paradigma festgelegt ist.

Insgesamt haben sich die eingesetzten Komponenten der Systemplattform dabei als einfach nutzbare und zuverlässige Grundlage für die Entwicklung und den Betrieb der Instant-Messenger-Plattform erwiesen. Dabei hat sich das gebildete Overlay-Netzwerk auch bei einer erhöhten Anzahl von Teilnehmern als unproblematisch erwiesen, sodass das Skalierungsverhalten ebenfalls positiv bewertet werden kann. Da die bezüglich der Netzwerktopologie gemachten Annahmen eines hybriden Aufbaus als typisch für Umgebungen anzusehen sind, in denen die kontextbasierte Ko-

operation besonders vorteilhaft eingesetzt werden kann, sind die erzielten positiven Ergebnisse entsprechend übertragbar.

7.3.2. Umsetzung klassischer kontextbasierter Aufgaben

Durch die Erweiterung der Kommunikationsplattform der Systemarchitektur um den Kontextdienst mit seinem generischen Kontextmodell und -verwaltungssystem, entsteht eine im Funktionsumfang klassische kontextbasierte Middleware. Deshalb wurde diese Ausbaustufe der Systemplattform daraufhin untersucht, inwieweit sie den Aufbau etablierter kontextbezogener Anwendungen erlaubt. Dazu wurde das Instant-Messenger-Thema wieder aufgegriffen und ein um kontextbasierte Funktionalität erweiterter Messaging Client umgesetzt (vgl. Abbildung 7.3).

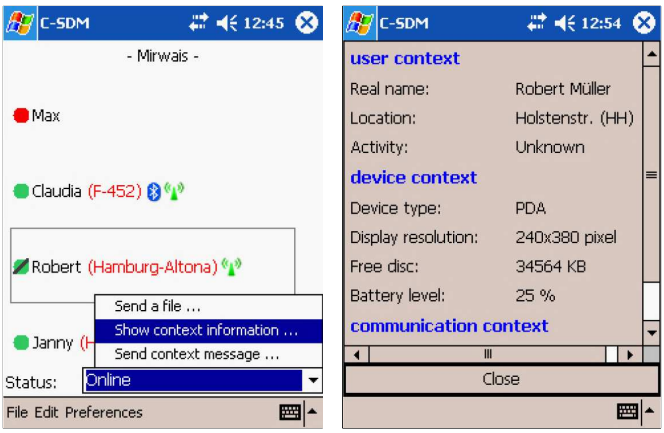


Abbildung 7.3.: Kontextbasierte Instant-Messenger-Anwendung (aus (Tur06))

Bei der Realisierung der Anwendung lag der Fokus dabei einerseits auf der Integration geeigneter Kontextausschnitte und zum anderen auf deren sinnvolle Einbindung in die Funktionalität eines Instant Messengers. Insgesamt wurde die Anwendung um folgende kontextbezogene Funktionalität erweitert (vgl. [Tur06]):

- *Erweiterung der Buddy-Liste um kontextbezogene Informationen*, wie dem Aufenthaltsort und die Netzwerkkonnektivität,
- Abruf *detaillierter Kontextinformationen* zu jedem Teilnehmer aus der Buddy-Liste,
- Versand *ortsbezogener Nachrichten*, die einem Teilnehmer angezeigt werden, sobald er einen vorbestimmten Ort erreicht und

- *Kontextbezogene Warnungen* beim Dateitransfer, bei denen Kontextinformationen, wie zum Beispiel der verfügbare Speicher, genutzt werden, um vorab zu bestimmen, ob ein Datentransfer sinnvoll ist.

Ziel dieser praktischen Evaluierung ist es zu zeigen, dass die Systemplattform nicht auf die Ausführung von Mobilen Prozessen beschränkt ist, sondern eine Weiterentwicklung klassischer kontextbezogener Systemplattformen darstellt. Deshalb müssen auch solche klassischen kontextbezogenen Aufgaben, wie die Bereitstellung und Aktualisierung von Kontextinformationen und deren Nutzung aus Anwendungen heraus, mit Hilfe der entwickelten Middleware umsetzbar sein.

Projektergebnisse Die Umsetzung eines kontextbasierten Instant Messengers hat exemplarisch gezeigt, dass die entwickelte Systemplattform dazu geeignet ist, die Implementierung klassischer kontextbezogener Anwendungen zu unterstützen. Dies ist deshalb notwendig, da die kontextbasierte Kooperation in Form von Mobilen Prozessen eine konsequente Weiterentwicklung bestehender Middleware-Ansätze darstellen soll und deshalb auch Anwendungen und Eigenschaften, die bereits bisher möglich sind, von der erweiterten Infrastruktur unterstützt werden müssen. Hierbei hat besonders der Ansatz der Trennung von Kontextmodell und -verwaltung sowie die Föderation lokaler Kontextausschnitte zu einem Gesamtkontext die Abbildung, Implementierung und Verwendung relevanter Kontextausschnitte vereinfacht. Dabei wurden die Dienste, die die Kontextinformationen auf den einzelnen Geräten bereitstellen, als Rohdatenquelle eingebunden. Hierauf aufbauend konnte ein Kontextmodell für den Instant-Messenger definiert werden und als Teilmodell dem Gesamtsystem hinzugefügt werden. In diesem Zusammenhang hat sich auch gezeigt, dass durch die generische Struktur des Kontextmodells Attribute und Entitäten leicht umsetzbar sind und auch in anderen Kontextausschnitten wieder verwendbar sind. Durch die individuell bestimmbaren Aktualisierungszeitpunkte der einzelnen Kontextattribute besteht eine besonders gute Möglichkeit, den Erwerb und die Aktualität von föderierten Teilen des relevanten Kontexts optimal an die Bedürfnisse der Anwendung anzupassen. So werden zum Beispiel Kontextangaben zum Aufenthaltsort wesentlich öfters aktualisiert als diejenigen Angaben über die zur Verfügung stehenden Speicherkapazitäten. Damit wird der Kommunikationsaufwand möglichst gering gehalten, da nicht das Attribut mit dem höchsten Aktualisierungsbedarf die Größe der Update-Intervalle bestimmt.

Insgesamt hat sich also damit gezeigt, dass klassische kontextbezogene Anwendungen durch die Systemplattform unterstützt werden. Hierbei ist durch den generischen Aufbau des Kontextbasismodells eine Basis geschaffen worden, um individuell an die Bedürfnisse unterschiedlicher Anwendungen angepasste Kontextausschnitte zu realisieren und föderieren. Damit hat sich die kontextbasierte Ausbaustufe der Systemplattform als geeignete Basis für die Erweiterung zu einer Middleware zur Ausführung mobiler Prozesse herausgestellt.

7.3.3. Einsatz mobiler Prozesse zur Realisierung komplexer Aufgaben

Die praktische Evaluierung der Gesamtplattform zur Ausführung mobiler Prozesse basiert auf einem Beispielszenario und hat zum Ziel, die mit der kontextbasierten Kooperation verbundenen Annahmen und Erwartungen zu überprüfen. Dies sind zum einen die generelle Umsetzbarkeit des Konzepts Mobiler Prozesse und zum anderen die Verdeutlichung der Nutzensteigerung für den einzelnen Teilnehmer einer mobilen Umgebung durch das Ausschöpfen größerer Teile des Gesamtpotenzials der Umgebung.

Das für die Evaluierung umgesetzte Beispielszenario ist aus dem Umkreis eines Journalisten entnommen, der für seinen Arbeitgeber live von unterschiedlichen und geografisch verteilten Großereignissen berichtet. Dabei umfasst dieses Szenario alle wesentlichen Eigenschaften und Anforderungen, die erst durch die Verwendung von Mobilen Prozessen ermöglicht werden. Das Szenario beinhaltet hierbei unterschiedliche Netzwerkarten, integriert unterschiedlich leistungsfähige Geräte, die erst durch ihre Kooperation eine abstrakte Aufgabe erfüllen können, und zeigt die Notwendigkeit der Integration nicht-funktionaler Aspekte bei der Ausführung des Mobilen Prozesses auf. Im Detail sieht das Szenario folgendermaßen aus (vgl. Abbildung 7.4):

Der Journalist merkt, als er um 11:00 Uhr am Fughafen ankommt, dass er seine Akkreditierung für das Ereignis, über das er berichten soll, vergessen hat. Da er jedoch nur bis 12:00 Uhr seine Akkreditierung über den Server des Veranstalters erneut abrufen kann, muss er unverzüglich den Abruf des Dokuments vornehmen und kann nicht warten, bis er im Hotel eintrifft. Deshalb verwendet er seinen PDA, um einen Mobilen Prozess zu initiieren, der das Dokument abruft und ausdruckt (1). Weil der PDA jedoch aufgrund seiner technischen Limitierung und der fehlenden direkten Verbindung zum Server des Veranstalters nicht in der Lage ist, den Web-Service zum Erneuern

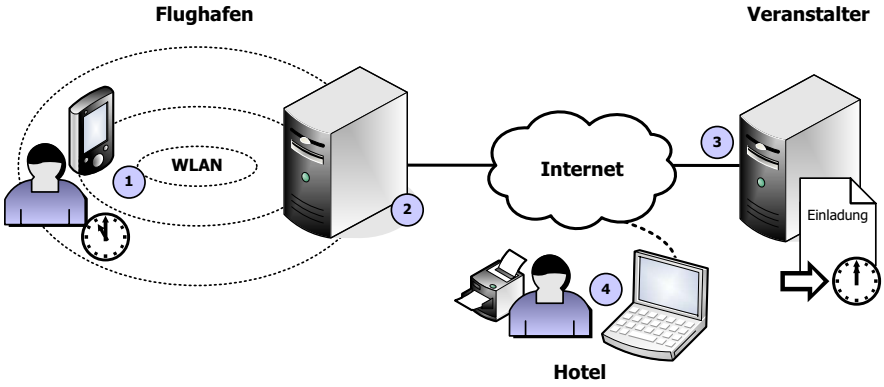


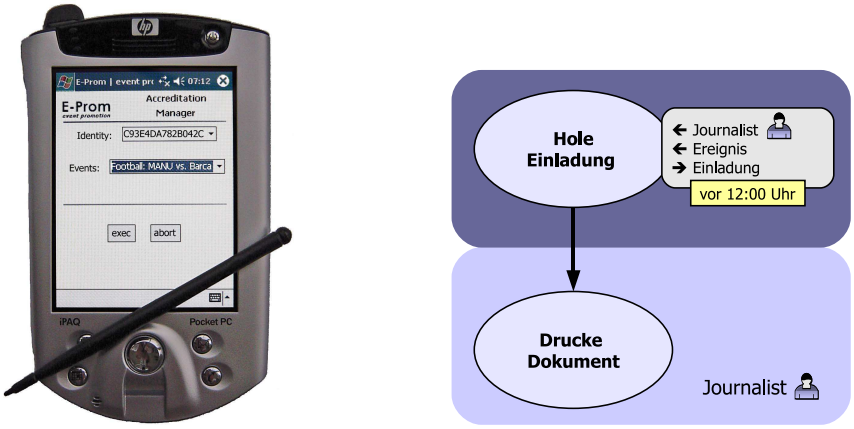
Abbildung 7.4.: Beispielszenario: Journalist

der Akkreditierung direkt aufzurufen, muss der PDA mit anderen Geräten in seiner Umgebung zusammen arbeiten, damit das Dokument rechtzeitig abgerufen werden kann. Der mobile Prozess muss also auf ein Gerät migrieren, das den Web-Service des Veranstalters aufrufen kann, und dann zum Journalisten zurückkehren, damit das Dokument bei ihm ausgedruckt werden kann. Da sich der Journalist noch am Flughafen aufhält, migriert der Prozess und damit die Ausführungsverantwortung auf den über WLAN erreichbaren Server des Flughafens (2). Dieser kann, da er mit dem Internet verbunden ist, den Web-Service des Veranstalters erreichen und so das Abrufen des Dokuments veranlassen (3). Das als Ergebnis dieses Aufrufs übermittelte Dokument fügt der Flughafen-Server dem Mobilen Prozess hinzu und hat somit die erste Teilaufgabe des Mobilen Prozesses abgearbeitet.

Die nächste Teilaufgabe des Mobilen Prozesses, also das Drucken des Akkreditierungsdokuments, beinhaltet die Rückkehr des Mobilen Prozesses zum Journalisten. Deshalb enthält der mobile Prozess eine Referenz auf die digitale Identität des Journalisten als nicht-funktionaler Parameter der Teilfunktionalität „Drucken“. Da der PDA, selbst wenn er vom Flughafen-Server aus noch erreichbar wäre, den Mobilen Prozess mit dem umfangreichen Akkreditierungsdokument aufgrund seiner begrenzten Speicherkapazität nicht wieder aufnehmen kann, muss der Flughafen-Server solange mit der Migration des Prozesses warten, bis ein Gerät mit der digitalen Identität des Journalisten und genügend Speicher verfügbar ist. Dieser Migrationsschritt erfolgt hier, sobald der Journalist im Hotel angekommen ist und sich mit seinem Laptop mit dem Internet verbindet, da dieses Gerät mit der gültigen Identität assoziiert ist und den Prozess aufnehmen kann (4). Nun ist es

dem Journalisten möglich, einen Drucker in seiner Nähe auszusuchen, also zum Beispiel den in der Rezeption des Hotels, und er erhält das benötigte Akkreditierungsdokument durch die kontextbasierte Kooperation seines PDAs, des Flughafen-Servers und seines Laptops.

Zur Realisierung des Szenarios wurde die Beispielanwendung *Accreditation Manager* geschrieben, die auf der Basis eines Prozess-Templates und den vom Benutzer gemachten Angaben einen gültigen Mobilen Prozess auf einem PDA erzeugt und der DEMAC-Middleware zur Ausführung übergibt. Dabei dient die Beispielanwendung lediglich der Parametrisierung des Template-Prozesses und führt selbst keine Aktivitäten des Prozesses aus (vgl. Abbildung 7.5(a)).



(a) Beispielanwendung (b) Prozessaufbau

Abbildung 7.5.: Umsetzung des Beispielszenarios

Das Prozess-Template besteht dabei aus den beiden im Szenario erwähnten Teilaktivitäten *„Hole Einladung“* und *„Drucke Dokument“*, die sequenziell durchlaufen werden (vgl. Abbildung 7.5(b)). Als Parameterdaten des Prozesses werden durch den Journalisten mittels des *Accreditation Manager* seine digitale Identität und das Ereignis, bei dem er seine Akkreditierung erneuern möchte, an den Mobilen Prozess gebunden. Zusätzlich wird die zeitliche Restriktion, die besagt, dass das Dokument bis um 12:00 Uhr abgerufen sein muss, während der Instanziierung des Mobilen Prozesses automatisch durch das Anwendungsprogramm hinzugefügt. Hierbei können Variablen, die Parameterdaten enthalten, sowohl als Eingabe- beziehungsweise Ausgabeparameter von Aktivitäten dienen als auch zur Definition von nicht-funktionalen Rahmenbedingungen im Mobilen Prozess. Dies wird, zum Beispiel durch die Verwendung der digitalen Identität des

Journalisten einmal als Eingabedatum der Aktivität „*Hole Einladung*“ und zum anderen als nicht-funktionaler Parameter zur Steuerung der Migration des Prozesses zurück zum Journalisten deutlich.

Für die beiden Aktivitäten dieses Prozesses sind entsprechende abstrakte Dienstklassen definiert und implementiert worden. Dabei ist der Dienst zum Erneuern der Akkreditierung über eine spezifische Dienstklasse des Veranstalters definiert, der Druckdienst hingegen ist eine Implementierung eines allgemeinen Druckdienstes, der beliebige XML-kodierte Dokumente drucken kann und somit generisch für unterschiedliche mobile Prozesse eingesetzt werden kann. Eine konkrete Instanz eines Mobilen Prozesses, der beispielsweise durch den *Accreditation Manager* erzeugt wurde, findet sich in Anhang B.

Implementiert wurde das Beispielszenario zur Erneuerung der Akkreditierung eines Journalisten auf der Basis der *Java-Plattform, Micro Edition* und der *IBM J9* als Ausführungsumgebung. Um die Migration über den nicht-funktionalen Identitätsparameter automatisch steuern zu können, wurde ein entsprechendes Kontextmodell entworfen, das die digitale Identität des Journalisten aufnehmen kann. Dieses wurde sowohl bei dem den Mobilen Prozess initiierenden Gerät als auch bei dem Gerät, das den Prozess empfangen soll, implementiert. Ein drittes Gerät war hingegen als Einziges in der Lage, den Akkreditierungsdienst auszuführen. Alle Geräte wurden dabei nicht explizit im Mobilen Prozess hinterlegt, sondern wurden allein aufgrund ihrer nicht-funktionalen Eigenschaften durch die Middleware zur Ausführung der Aktivitäten ausgesucht und eingebunden.

Projektergebnisse Die Evaluierung der Gesamtarchitektur anhand des Beispielszenarios hat – auch auf die Allgemeinheit übertragbar – gezeigt, dass die Erwartungen, die mit dem Konzept der kontextbasierten Kooperation und ihrer Umsetzung als mobile Prozesse verbunden waren, erfüllt wurden. Dabei ist deutlich geworden, dass durch die Kooperation der Geräte einer heterogenen, mobilen Umgebung dem mobilen Nutzer komplexe Aufgaben auch über leistungsschwache Geräte zur Verfügung stehen, die zuvor mittels dieser Geräte nicht ausführbar waren. Durch das Einbeziehen von Kontextinformationen zur Steuerung der Migration sind dabei die analytisch gewonnenen Ausführungswahrscheinlichkeiten sogar noch übertroffen worden, sodass die stochastisch bestimmten Grenzausführungswahrscheinlichkeiten als untere Schranke für die tatsächlichen Ausführungswahrscheinlichkeiten mit Kontextbezug angesehen werden können. Dabei gilt natürlich, dass es im Einzelfall auch vorkommen kann,

dass kein geeigneter Migrationspartner innerhalb der nicht-funktionalen Zeitrestriktionen gefunden wird und deshalb die Ausführung des Mobilen Prozesses scheitert. Es kann also keine Ausführungsgarantie gegeben werden, jedoch ist die Wahrscheinlichkeit, dass die Aktivitäten des Mobilen Prozesses erfolgreich ausgeführt werden insgesamt höher als wenn die Ausführung der Aktivitäten nicht versucht worden wäre. Dieses Beispielszenario führt also zu der Erkenntnis, dass Mobile Prozesse nur dann verwendet werden dürfen, wenn die Ausführung der einzelnen Aktivitäten des Mobilen Prozesses in der Umgebung garantiert werden kann oder das Scheitern des Prozesses als ein mögliches Ergebnis der Ausführung akzeptabel ist.

Insgesamt bestätigt sich also, dass durch die kontextbasierte Kooperation in Form von Mobilen Prozessen eine Steigerung der Ausnutzung des Gesamtpotenzials der (mobilen) Umgebung stattfindet. Zudem ist die Umsetzung des Konzepts durch die abstrakte Definition der Mobilen Prozesse sehr flexibel und bietet einen generischen Ansatz zur Definition der komplexen Aufgaben und deren Ausführung in einer heterogenen Anwendungslandschaft. Einen wesentlichen Anteil daran hat auch das eingeführte Konzept der abstrakten Dienstklassen, da nur durch sie eine technologieübergreifende Definition der Aufgaben möglich wird. Damit stellt der entwickelte Middleware-Ansatz eine echte Erweiterung etablierter kontextbasierter Systeme dar.

8. Zusammenfassung und Ausblick

Die nachfolgenden Abschnitte fassen als Abschluss der Arbeit die Ziele sowie die erzielten Ergebnisse noch einmal zusammen und geben einen Ausblick über mögliche Ergänzungen und weitere Fragestellungen. Hierbei wird zunächst eine Zusammenfassung der Motivation und der Zielsetzung gegeben sowie auf die im Verlauf der Arbeit herausgearbeiteten Resultate eingegangen. Diese Darstellungen bilden dann die Basis für den abschließenden Ausblick auf mögliche Perspektiven und weitere Ansatzpunkte, die sich aus der Arbeit ergeben.

8.1. Zusammenfassung

Die vorliegende Arbeit dokumentiert, dass das Gebiet des Mobile Computing zwar einerseits eine zentrale Rolle bei der Weiterentwicklung moderner Informations- und Kommunikationstechnologien einnimmt, aber andererseits zurzeit noch Defizite bei der Unterstützung der Ausführung ad-hoc entstehender, langlebiger und komplexer Aufgaben bestehen. Hierbei wurde aufgezeigt, dass Anforderungen, die durch die Mobilität entstehen, zusätzliche und erweiterte Bedürfnisse und Einschränkungen begründen, die sich zum Teil deutlich von den etablierten Methoden und Verfahren traditioneller Verteilter Systeme abgrenzen. Diese Erkenntnisse haben zwar zu einer Reihe von unterstützenden Systemplattformen für das Mobile Computing geführt, die jedoch durch ihren oftmals strikten Anwendungsbezug die Potenziale, die eine dynamische und heterogene Systemumgebung bietet und die bei der Ausführung komplexer Aufgaben nutzbringend wären, nur in einem geringen Grad auch tatsächlich nutzbar machen.

Aus dieser Beobachtung leitete sich dementsprechend auch das übergeordnete Ziel der Arbeit ab: Es waren Konzepte und Techniken zu entwickelt, die es durch die Zusammenführung der Potenziale der an einer mobilen Umgebung teilnehmenden Systeme ermöglichen, auch bisher nicht realisierbare ad-hoc entstehende und komplexe Aufgaben auszuführen. Dabei war diese Unterstützung durch die konsequente Weiterentwicklung bestehender (kontextbezogener) Ansätze hin zu kooperativen Systemstrukturen für mobile Umgebungen umzusetzen. Konkret manifestierte sich dieses Ziel in der Entwicklung einer entsprechenden System-

plattform mit konzeptionellen und technischen Anteilen, die eine kontextbasierte Kooperation durch die Migration und verteilte Ausführung von Prozessen im Umfeld mobiler Systeme bietet.

Das Fundament und der Ausgangspunkt für die Entwicklung von Konzepten und Technologien zur Umsetzung einer solchen kontextbasierten und kooperativen Systemplattform bildeten dabei Untersuchungen zum Mobilitätsbegriff, dessen Einfluss und Umsetzung in aktuellen Middleware-Plattformen im Mobile Computing sowie zur metasprachlichen Beschreibung komplexer Aufgaben. Hierzu wurde in Kapitel 2 der vorliegenden Arbeit in die unterschiedlichen Klassen der Mobilität, die in Informationssystemen auftreten können, eingeführt und es wurden mobile von traditionellen Verteilten Systemen abgegrenzt. Dabei wurde zudem auf die intrinsischen Eigenschaften mobiler Systeme eingegangen, die einen festen Rahmen bildeten, der bei der Entwicklung der konzeptionellen Modelle und der technologischen Umsetzungen zu beachten war. Zudem wurden die für mobile Systeme charakteristischen Kommunikationsmodelle sowie die sich hieraus ergebenden inhärenten Sicherheits- und Vertrauensproblematiken mobiler Umgebungen aufgezeigt. Hierauf aufbauend wurde in Kapitel 3 untersucht, welche Anforderungen an Middleware-Plattformen zur Unterstützung mobiler Systeme und Anwendungen bestehen und wie diese in existierenden Konzepten und Plattformen umgesetzt werden. Vor diesem Hintergrund wurde in die beiden großen Klassen von kommunikationsorientierten und kontextbezogenen Middleware-Ansätzen eingeführt, und es wurden prominente Vertreter einzelner konzeptioneller Realisierungsansätze beschrieben. Kapitel 4 schloß den grundlegenden Abschnitt der Arbeit mit einer Betrachtung ab, wie durch Dienste und Prozesse komplexe Aufgaben metasprachlich beschrieben werden können und so eine plattformunabhängige Definition von Aufgaben in einer dynamischen und heterogenen mobilen Umgebung möglich ist. Dabei wurde auch auf die Grundlagen der Dienstorientierung im Rahmen der Service-Oriented Architecture eingegangen und es wurde herausgearbeitet, wie einzelne Dienste zu dienstübergreifenden (und zum Teil auch langlebigen) Prozessen komponiert werden können. Zudem wurde aufgezeigt, welche Lebenszyklen bei der Ausführung von Prozessen durchlaufen werden und welche unterschiedlichen Ansätze zur Komposition von Prozessen existieren und wie die entsprechenden Paradigmen zur Orchestrierung und Choreographie von Diensten in etablierten Modellierungssprachen umgesetzt werden.

Aufbauend auf den zuvor herausgearbeiteten, eher grundlegenden Aus-

sagen und Überlegungen wurde in Kapitel 5 das konzeptionelle Modell der kontextbasierten Kooperation und seiner Realisierung in Form von Mobilen Prozessen erarbeitet. Hierzu wurde zunächst dargelegt, dass die bestehenden kommunikations- und anwendungsorientierten Middleware-Klassen für die Ausführung komplexer und benutzerzentrischer Aufgaben nicht ausreichen und deshalb eine erweiterte Klasse benutzerorientierter Middleware-Plattformen zu etablieren ist. Dabei wurde die Benutzerorientierung als konsequente Weiterentwicklung bestehender Ansätze verstanden, welche eine erhöhte Generalität und Anpassungsfähigkeit der Systeme hin zur Unterstützung a priori unbekannter und ad-hoc auftretender Aufgaben und Anwendungen ermöglicht. Hierbei wurde die oftmals in einer mobilen Umgebung bestehende Heterogenität und Dynamik nicht nur als Herausforderung angesehen, die eine Awareness und Adaptability einzelner Anwendungen nach sich zieht, sondern vielmehr als das eigentliche Potenzial, das eine mobile Umgebung dem Benutzer zur Ausführung seiner Aufgaben anbietet. Hierauf aufbauend wurde die kontextbasierte Kooperation eingeführt, welche die technologische Isolation mobiler Geräte aufhebt und so die Zusammenarbeit kontextbezogener Systeme ermöglicht. Damit wurde eine partnerschaftliche Verbindung zwischen allen beteiligten Systemen und Geräten einer mobilen Umgebung hergestellt, die große Teile der enthaltenen Dienste und Ressourcen auch anderen zugänglich macht.

Um dieses Modell auch unter den Rahmenbedingungen einer unzuverlässigen und sich meist stetig verändernden Umgebung zu realisieren, wurde dann das auf der kontextbasierten Kooperation aufbauende Konzept der Mobilen Prozesse eingeführt. Derartige Prozesse schaffen es durch eine lokale und temporär zentrale Ausführungsstrategie, komplexe Aufgaben in Form von Prozessen auch in sehr dynamischen Systemen kooperativ auszuführen. Dabei wurde unter anderem auch aufgezeigt, dass durch die eingeführte Migration der Prozesse, welche die langlebigen und komplexen Aufgaben im System repräsentieren, eine deutliche Steigerung der Ausführungswahrscheinlichkeit der einzelnen Aufgaben erreicht werden kann. Dabei konnten die durch dienstorientierte Architekturen etablierten Methoden und Verfahren zur Ausführung von Prozessen aber nicht uneingeschränkt übernommen werden. Es musste vielmehr ein erweiterter Prozesslebenszyklus etabliert werden, der die Migration der Prozesse integriert und somit die Ausführung Mobiler Prozesse erst ermöglicht. Zunächst musste hierbei konzeptionell das Einbinden von nicht-funktionalen Aspekten integriert werden, damit während der Ausführung von Mobilen Prozessen die Restriktionen, die der Initiator

über die reine Funktionalität hinaus mit dem Prozess verbindet, auch auf entfernten Ausführungseinheiten eingehalten werden können. Um solche Mobilen Prozesse umsetzen zu können, wurde ein entsprechendes konzeptionelles Prozessmodell entwickelt. Um dabei die Teilaufgaben eines Mobilen Prozesses (Aktivitäten) auf abstrakte und technologieunabhängige Weise beschreiben zu können, wurde zudem das Konzept der abstrakten Dienstklasse eingeführt, das als Weiterentwicklung der etablierten Konzepte zur Schnittstellenbeschreibung in dienstorientierten Systemen anzusehen ist. Da durch die Migration ein festes und auf einzelne Anwendungen beziehungsweise Anwendungsszenarien abgestimmtes Kontextmodell nicht sinnvoll ist, wurde auch ein abstraktes und generisches Kontextmodell und -managementsystem etabliert, das zur Steuerung der Ausführung Mobiler Prozesse und damit komplexer Aufgaben geeignet ist. Hierbei wurde ein auf abstrakte Dienstklassen abgestimmtes System zum föderierten Auffinden und Ansprechen von lokalen und entfernten Diensten etabliert, das die Anforderungen, die aus der Ausführung Mobiler Prozesse erwachsen, abdeckt und sich in das erweiterte Kontextsystem nahtlos eingliedert. Abschließend führte das Kapitel in an die Bedürfnisse und Eigenschaften Mobiler Prozesse angepasste Transaktionsmodelle ein, die auch unter den – aus Sicht der Transaktionsverwaltung schwierigen – Randbedingungen eines Systems zur verteilten Ausführung von Prozessen in einer dynamischen und unzuverlässigen Umgebung einen Transaktionsschutz für eine Vielzahl von Anwendungsszenarien ermöglicht.

Die praktische Umsetzbarkeit der kontextbasierten Kooperation in Form von Mobilen Prozessen wurde in Kapitel 6 nachgewiesen. Hierzu wurde auf die prototypische Umsetzung der eingeführten Konzepte eingegangen, die im Projekt DEMAC (Distributed Environment for Mobility-Aware Computing) realisiert wurden. Dabei wurde unter anderem auch auf die funktionalen Komponenten des Kommunikationssystems, des erweiterten Kontextdiensts und des Prozessdiensts eingegangen. Dabei realisiert der nachrichtenorientierte Transportdienst ein Overlay-Netzwerk, durch das unterschiedliche Transportprotokolle und Netzwerktechnologien zu einer homogenen Transportschicht für die darauf aufbauenden Dienste der DEMAC-Plattform vereinheitlicht werden. Aufgrund dieses Ansatzes zur Kapselung der heterogenen Kommunikationstechnologien in einer mobilen Umgebung mussten für den Aufbau und Betrieb des Overlay-Netzwerks unabhängige Adressierungs- und Routing-Schemata sowie Sicherheitskomponenten integriert werden. Auf dieser Basis wurde zudem ein Dienst zur ereignisbasierten Kommunikation eingeführt, so-

dass sowohl aktive als auch passive asynchrone Kommunikationsparadigmen durch die entwickelte Middleware realisiert werden. Damit schafft das Kommunikationssystem die Voraussetzung für die Offenheit der gesamten Systeminfrastruktur. Der Kontextdienst der vorgestellten Systemplattform setzt das abstrakte und damit anwendungsunabhängige Kontextmodell sowie die Managementsysteme um und integriert den erweiterten Verzeichnisdienst. Damit stellt diese Komponente eine wichtige Wissensbasis zur Realisierung der kontextbasierten Kooperation in Form von Mobilen Prozessen dar. Denn durch sie ist es erst möglich, funktionale und nicht-funktionale Anforderungen der technologieneutralen Mobilen Prozesse in konkrete Dienste und Ausprägungen umzusetzen. Dabei wurden auch an dieser Stelle die intrinsischen Eigenschaften mobiler Systeme berücksichtigt, indem zum Beispiel durch föderierte Ansätze und Übersetzungskomponenten die Dynamik und Heterogenität mobiler Umgebungen in die Umsetzungskonzepte einflossen. Zur eigentlichen Ausführung Mobiler Prozesse dient der Prozessdienst der Systemplattform. Dieser bildet eine Ausführungsumgebung, die den erweiterten Prozesslebenszyklus abbildet und die abstrakte Beschreibung der Prozesse mit Hilfe des Kommunikationssystems und des Kontextdiensts in einzelne lokale oder entfernte Dienstaufrufe umsetzt. Dabei wurde der Prozessdienst modular gestaltet, sodass er sich der unterschiedlichen Leistungsfähigkeit einzelner Geräte anpassen kann. Somit ist es sogar Geräten, die Mobile Prozesse selbst nicht ausführen können, möglich, komplexe Aufgaben anzustoßen, die durch die Kooperation anderer Geräte dennoch ausgeführt werden können. Zudem kann durch die Anbindung von Erweiterungsmodulen die Leistungsfähigkeit des Prozessdiensts weiter gesteigert werden, was am Beispiel der Transaktionsunterstützung exemplarisch gezeigt wurde. Damit stellt die DEMAC-Systemplattform eine prototypische Implementierung dar, mit deren Hilfe heterogene (mobile) Geräte eine Umgebung zur Realisierung der kontextbasierte Kooperation aufbauen können.

In Kapitel 7 wurden abschließend die eingeführten Konzepte und die prototypische Implementierung einer exemplarischen Evaluierung unterzogen. Dabei wurden die eingeführten Konzepte zunächst anhand des erarbeiteten Anforderungskatalogs bewertet und anschließend die Systemplattform durch Beispielanwendungen und exemplarische Szenarios praktisch untersucht. Hierbei zeigte sich, dass die Systemplattform die geforderten Eigenschaften abdeckt und die durch die Einführung der kontextbasierten Kooperation erwartete Steigerung der Ausführungswahrscheinlichkeit im Mittel auch tatsächlich eintritt. Zudem wurde hier gezeigt, dass

mit den erweiterten Konzepten und deren prototypischen Umsetzungen in der DEMAC-Systemplattform sowohl etablierte Anwendungen umgesetzt werden können, als auch bisher nicht mögliche ad-hoc entstehende, langlebige und komplexe Aufgaben. Als Ergebnis dieser Arbeit kann festgehalten werden, dass durch das Etablieren der kontextbasierten Kooperation ein weiterer Schritt hin zur Realisierung der Vision des Ubiquitous Computing gelungen ist. Durch das Konzept der Mobilen Prozesse und deren Umsetzung in Form einer prototypischen Systemplattform werden die Potenziale mobiler Umgebungen in einem höheren Umfang für einzelne Aufgaben und Anwendungen verfügbar gemacht. Dabei können nahezu beliebige Geräte dazu verwendet werden, komplexe Aufgaben zu initiieren, die dann automatisch unter den Rahmenbedingungen des Initiators ausgeführt werden und so den Nutzern einen intuitiveren sowie unaufdringlicheren Zugang zum verteilte System (mobiler) Geräte bietet. Durch das Einbeziehen von Kontextwissen und von föderativen Konzepten ist damit ein flexibles System entstanden, das für die Ausführung der Aufgaben, die an es gestellt werden, eine auf den verfügbaren Ressourcen basierende (verteilte) Ausführung organisiert. Hierbei wird es durch die Migration der Prozesse zwischen unterschiedlichen Geräten der Umgebung ermöglicht, diese aus wechselnden Blickwinkeln zu betrachten und so auch die Dienste und Ressourcen des Systems mit einzubeziehen, die bei einer von nur einem Gerät zentral gesteuerten Ausführung der Prozesse, zum Beispiel durch technologische Unzulänglichkeiten, ansonsten verborgen geblieben wären.

8.2. Ausblick

Abschließend wird in diesem Ausblick auf zukünftige Entwicklungs- und Forschungsmöglichkeiten im Umfeld der in dieser Arbeit betrachteten Erweiterung von Middleware-Systemen im Mobile Computing eingegangen. Dabei werden nicht nur die Fragestellungen betrachtet, die sich direkt aus dieser Arbeit ergebende, sondern auch auf Forschungstendenzen, die den behandelten thematischen Hintergrund aufgreifen.

Die Relevanz der Integration von Prozessen in Form von Workflows in kontextbezogene Systemplattformen als wesentlicher Schritt hin zu einer neuen Generation von Middleware-Plattformen hat sich in ähnlicher Form ebenfalls im Projekt Nexus (vgl. Abschnitt 3.3.3.2) herausgestellt. Deshalb wird auch in diesem Sonderforschungsbereich der Deutschen Forschungsgemeinschaft (DFG) in der aktuell begonnenen zweiten Förderperiode ein neues Teilprojekt (E1) eingeführt, dass sich mit der Entwicklung von ele-

mentaren Konzepten und Methoden zur Unterstützung „kontextbezogener Workflows“ beschäftigt. In ein derartiges Nexus-Teilprojekt könnten die Ergebnisse der vorliegenden Arbeit – insbesondere die, die sich auf das zweistufige Programmieren von lang andauernden und komplexen kontextbezogenen Aufgaben beziehen – nahezu unmittelbar mit einfließen. Wesentliche Teile der dort skizzierten Problematiken wurden bereits durch die in der vorliegenden Arbeit eingeführte kontextbasierte Kooperation in Form vom Mobilen Prozessen untersucht, sodass entsprechende konzeptionelle Lösungsvorschläge aus den hier erarbeiteten Ergebnissen abgeleitet werden könnten. [LNW08, WKNL07].

Die hier eingeführten Verfahren und Methoden zur Realisierung einer Systemplattform zur Ausführung von Mobilen Prozessen – und damit zur Umsetzung des Ansatzes der kontextbasierten Kooperation – stellen eine ausbaufähige Systemplattform bereit, um darauf noch erweiterte und weiter optimierte Konzepte zu entwickeln. So können zum Beispiel auf der Ebene des Aufbaus und der Verwaltung des grundlegenden Overlay-Netzwerks optimierte Strategien zum Nachrichten-Routing betrachtet werden. Hierbei bietet sich die Möglichkeit, durch das Einführen von unterschiedlichen Rollen, wie Konzentratoren und Gateway-Knoten, bei ausgedehnten Teilnetzen ein verbessertes Skalierungsverhalten und eine geringere Kommunikationslast zu erhalten. Zudem bietet es sich durch die bestehende Modularisierung an, die Awareness auf die Anforderungen der von einem System verarbeiteten Mobilen Prozesse auszudehnen und durch entsprechende dynamische Rekonfigurierung der aktiven Module eine für die Prozesse optimal abgestimmte Ausführungsumgebung zu schaffen und gleichzeitig möglichst effizient mit den bestehenden Ressourcen des Systems umzugehen. Dabei ist jedoch der Overhead der Rekonfigurierung in das Verhältnis zu den frei werdenden Ressourcen zu setzen und so abzuwägen, ob hier weitere Potenziale zur Entwicklung optimierter Konfigurationsstrategien bestehen.

Da durch die verteilte Ausführung von Mobilen Prozessen eine Entkopplung der Anwendungslogik von der Darstellung einhergeht, bietet es sich zudem an, auch die Interaktion zwischen Nutzern und Prozessen durch die abstrakte Definition von generischen Interaktionsprozessen anstelle von vordefinierten Ein- und Ausgabediensten zu realisieren. Durch einen solchen Ansatz erhöht sich die Flexibilität und die Einsatzbreite der Mobilen Prozesse deutlich, da zur Laufzeit die notwendige Kommunikation zwischen Nutzer und Prozess auf vielfältige Weise umgesetzt werden kann. So können Interaktionsprozesse durch entsprechende Interpreter spontan

vom jeweiligen System in unterschiedliche Repräsentationsformen – wie zum Beispiel grafische oder sprachgesteuerte Bedienoberflächen – überführt werden und so die Bandbreite der zur Ausführung von Interaktionsaktivitäten eines Prozesses zur Verfügung stehenden Systeme deutlich erhöhen.

Wie in der Evaluierung der Umsetzung des Konzepts der Mobilen Prozesse bereits ausgeführt (vgl. Abschnitt 7.1), bieten schließlich die bisher eingeführten Sicherheitskomponenten lediglich einen Basisschutz auf Kommunikationsebene. Dementsprechend bieten sich hier Potenziale zur Erweiterung des eingeführten Ansatzes um breitere Konzepte zur Sicherstellung weitergehender Sicherheits- und Vertrauensaspekte an. Als Ergänzung ist so zum Beispiel die Integration eines digitalen Identitätsmanagements denkbar, das auf der Basis von Vertrauensmetriken und dem (sozialen) Kontext einen differenzierten Zugriff auf die vom Gerät bereitgestellten Dienste und Kontextinformationen ermöglicht. Hierbei lässt sich auf Erfahrungen und Konzepte zurückgreifen, die zum Beispiel im Projekt onefC (open network environment for Citizens, vgl. [BK04a, BK04b, BZL03]) erarbeitet wurden. Diese müssen jedoch noch an die Bedürfnisse und Eigenschaften Mobiler Prozesse sowie dynamischer und heterogener Systemumgebungen angepasst werden. Zusammen mit den nicht-funktionalen Parametern Mobiler Prozesse bieten sie dann ein gesteigertes Potenzial zur effektiven Steuerung und Umsetzung der Schutzbedürfnisse der an derartigen Prozessen beteiligten Komponenten.

Ein weiterer interessanter Ansatz ist auch die Verteilung einzelner Module der Systemplattform zur Ausführung Mobiler Prozesse in der mobilen Umgebung selbst. So können zum Beispiel Dienste – wie das Diensteverzeichnis zur Auflösung von abstrakten Dienstklassen in konkrete Dienstinstanzen – durch leistungsfähige Geräte anderen Systemteilnehmern als Proxy-Dienste angeboten werden, sodass auch Geräte ohne eigenes Diensteverzeichnis und lokale Dienstinstanzen Mobile Prozesse zumindest durch entfernte Dienstaufrufe voranbringen können. Hierbei muss jedoch Rücksicht auf die Verfügbarkeit der Proxy-Systeme sowie auf eine abstrakte Normierung der entsprechenden Verzeichnisschnittstelle genommen werden.

Abschließend sollen die Möglichkeiten der Integration von Mobilen Prozessen in bestehende Workflow-Umgebungen angesprochen werden. Gerade die Integration von mobilen Anteilen eines Workflows in bestehende Unternehmensprozesse ist sicherlich ein naheliegender Erweiterungsschritt, da so das Einsatzgebiet Mobiler Prozesse wesentlich erweitert wer-

den kann. Damit können Mobile Prozesse, da sie eine zustandsbehaftete Erweiterung der abstrakten Prozessaustauschsprache XPDL sind, nicht nur zum Austausch von Prozessdefinitionen verwendet werden, sondern es können sogar laufende Prozessinstanzen mit ihrem jeweiligen Ausführungszustand ausgetauscht werden. So kann zum Beispiel eine Prozessinstanz in Fehlersituationen oder zu Optimierungszwecken von einem Produktivsystem auf ein Ersatzsystem migrieren, ohne dass die Ausführung der Prozessinstanz abgebrochen werden muss. Dies bietet gerade bei produktionsbegleitenden Workflows, die materielle Auswirkungen haben können, den Vorteil, dass bereits geleistete Arbeit erhalten bleibt.

Insgesamt bieten sich also noch eine Reihe weiterer elementarer und interessanter Entwicklungsmöglichkeiten auf Basis der kontextbasierten Kooperation und Mobilen Prozesse an. Diese werden unter anderem auch weiterhin im Umfeld des Projekts DEMAC vorangetrieben, sodass mit Abschluss dieser Arbeit lediglich ein erstes Fundament entstanden ist, das in Zukunft weiter ausgebaut werden soll.

A. Untersuchungsergebnisse zur Ausführungswahrscheinlichkeit

Hop	Migration 20%				Migration 80%			
0	20,00%	40,00%	60,00%	80,00%	20,00%	40,00%	60,00%	80,00%
1	23,20%	44,80%	64,80%	83,20%	32,80%	59,20%	79,20%	92,80%
2	23,71%	45,38%	65,18%	83,33%	40,99%	68,42%	85,34%	94,85%
3	23,79%	45,45%	65,21%	83,33%	46,23%	72,84%	87,31%	95,18%
4	23,81%	45,45%	65,22%	83,33%	49,59%	74,96%	87,94%	95,23%
5	23,81%	45,45%	65,22%	83,33%	51,74%	75,98%	88,14%	95,24%
6	23,81%	45,45%	65,22%	83,33%	53,11%	76,47%	88,20%	95,24%
7	23,81%	45,45%	65,22%	83,33%	53,99%	76,71%	88,23%	95,24%
8	23,81%	45,45%	65,22%	83,33%	54,55%	76,82%	88,23%	95,24%
9	23,81%	45,45%	65,22%	83,33%	54,92%	76,87%	88,23%	95,24%
10	23,81%	45,45%	65,22%	83,33%	55,15%	76,90%	88,23%	95,24%
11	23,81%	45,45%	65,22%	83,33%	55,29%	76,91%	88,24%	95,24%
12	23,81%	45,45%	65,22%	83,33%	55,39%	76,92%	88,24%	95,24%
13	23,81%	45,45%	65,22%	83,33%	55,45%	76,92%	88,24%	95,24%
14	23,81%	45,45%	65,22%	83,33%	55,49%	76,92%	88,24%	95,24%
15	23,81%	45,45%	65,22%	83,33%	55,51%	76,92%	88,24%	95,24%
16	23,81%	45,45%	65,22%	83,33%	55,53%	76,92%	88,24%	95,24%
17	23,81%	45,45%	65,22%	83,33%	55,54%	76,92%	88,24%	95,24%
18	23,81%	45,45%	65,22%	83,33%	55,54%	76,92%	88,24%	95,24%
19	23,81%	45,45%	65,22%	83,33%	55,55%	76,92%	88,24%	95,24%
20	23,81%	45,45%	65,22%	83,33%	55,55%	76,92%	88,24%	95,24%

Tabelle A.1.: Analytische Ausführungswahrscheinlichkeiten

Hop	Analytisch	1. Lauf	2. Lauf	3. Lauf	Durchschnitt
0	40,00%	33,00%	46,00%	29,00%	36,00%
1	61,12%	58,00%	66,00%	55,00%	59,67%
2	72,27%	67,00%	81,00%	66,00%	71,33%
3	78,16%	73,00%	84,00%	70,00%	75,67%
4	81,27%	76,00%	86,00%	77,00%	79,67%
5	82,91%	80,00%	88,00%	78,00%	82,00%
6	83,78%	80,00%	89,00%	80,00%	83,00%
7	84,23%	83,00%	89,00%	81,00%	84,33%
8	84,48%	85,00%	89,00%	81,00%	85,00%
9	84,60%	86,00%	89,00%	82,00%	85,67%
10	84,67%	86,00%	89,00%	82,00%	85,67%
11	84,71%	87,00%	89,00%	82,00%	86,00%
12	84,72%	87,00%	89,00%	82,00%	86,00%
13	84,73%	88,00%	89,00%	84,00%	87,00%
14	84,74%	89,00%	89,00%	84,00%	87,33%
15	84,74%	90,00%	89,00%	84,00%	87,67%
16	84,74%	90,00%	89,00%	84,00%	87,67%
17	84,74%	91,00%	89,00%	84,00%	88,00%
18	84,75%	91,00%	89,00%	84,00%	88,00%

Tabelle A.2.: Experimentelle Ausführungswahrscheinlichkeiten

B. Prozessbeschreibung des Akkreditierungsprozesses

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Package
3    xmlns="[...]dpdl1.0.xsd"
4    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5    xsi:schemaLocation="[...]dpdl1.0.xsd"
6    Id="EPromRegistration" Name="EProm Szenario"
7    DPDLVersion="1.0">
8      <Participants>
9        <Participant Id="Journalist">
10         <Person>
11           520A5BB71A984775A94CB3AA49920148
12         </Person>
13       </Participant>
14     </Participants>
15     <Applications>
16     <Application Id="RegistrationService"
17       Name="EProm Registration Service">
18       <UUID>154C9B09E98148CF8C308A256D49A864</UUID>
19       <FormalParameters>
20         <FormalParameter Id="Identity" Index="1" Mode="IN">
21           <DataType>
22             <BasicType Type="STRING"/>
23           </DataType>
24         </FormalParameter>
25         <FormalParameter Id="Event" Index="2" Mode="IN">
26           <DataType>
27             <BasicType Type="STRING"/>
28           </DataType>
29         </FormalParameter>
30         <FormalParameter Id="XMLDocument" Index="3" Mode="OUT">
31           <DataType>
32             <BasicType Type="STRING"/>
33           </DataType>
34         </FormalParameter>
35       </FormalParameters>
36     </Application>
37     <Application Id="PrintService"
38       Name="XML-Document Print Service">
```

```
39      <UUID>154C9B09E98148CF8C308A256D49A864</UUID>
40      <FormalParameters>
41        <FormalParameter Id="XMLDocument" Index="1" Mode="IN">
42          <DataType>
43            <BasicType Type="STRING"/>
44          </DataType>
45        </FormalParameter>
46      </FormalParameters>
47    </Application>
48  </Applications>
49  <DataFields>
50    <DataField Id="IdentityField" Name="Identity">
51      <DataType>
52        <BasicType Type="STRING"/>
53      </DataType>
54      <InitialValue>Peter Smith</InitialValue>
55    </DataField>
56    <DataField Id="EventField" Name="Event">
57      <DataType>
58        <BasicType Type="STRING"/>
59      </DataType>
60      <InitialValue>
61        CL: Manchaster United vs. AS Roma
62      </InitialValue>
63    </DataField>
64    <DataField Id="XMLDocumentField" Name="XMLDocument">
65      <DataType>
66        <BasicType Type="STRING"/>
67      </DataType>
68      <InitialValue></InitialValue>
69    </DataField>
70  </DataFields>
71  <WorkflowProcess Id="1" Name="RegistrationProcess"
72    AccessLevel="PUBLIC" StartActivity="a1" InitActivity="a1"
73    LoggingRequired="true">
74    <ProcessHeader>
75      <Description>EProm Registration Process</Description>
76      <Priority>10</Priority>
77    </ProcessHeader>
78    <Activities>
79      <Activity Id="1" Name="Register4Event">
80        <Implementation>
81          <Tool ApplicationId="RegistrationService">
82            <ActualParameters>
```

```
            <ActualParameter>IdentityField</ActualParameter>
            <ActualParameter>EventField</ActualParameter>
            <ActualParameter>
              XMLDocumentField
            </ActualParameter>
          </ActualParameters>
        </Tool>
      </Implementation>
    </Activity>
    <Activity Id="2" Name="XML Print Service">
      <Implementation>
        <Tool ApplicationId="PrintService">
          <ActualParameters>
            <ActualParameter>
              XMLDocumentField
            </ActualParameter>
          </ActualParameters>
        </Tool>
      </Implementation>
    </Activity>
  </Activities>
  <ActivityRefs>
    <ActivityRef Id="a1" ActivityId="1" State="INACTIVE"/>
    <ActivityRef Id="a2" ActivityId="2" State="INACTIVE"
      ParticipantID="Jounalist"/>
  </ActivityRefs>
  <Transitions>
    <Transition Id="t1" Name="t1" From="a1" To="a2"/>
  </Transitions>
 </WorkflowProcess>
</Package>
```

Veröffentlichungen

Folgende Veröffentlichungen sind aus den Ergebnissen im Umfeld dieses Dissertationsprojektes hervorgegangen:

KUNZE, CHRISTIAN P., SONJA ZAPLATA, MIRWAIS TURJALEI und WINFRIED LAMERSDORF: *Enabling Context-based Cooperation: A Generic Context Model and Management System*. In: *Proceedings of the 11th International Conference on Business Information Systems (BIS 2008)*, Lecture Notes in Business Information Processing (LNBIP). Springer-Verlag, 2008. (Zur Veröffentlichung angenommen)

SONJA ZAPLATA und CHRISTIAN P. KUNZE, *Prozessmanagement im Mobile Computing - Kooperative Ausführung von Geschäftsprozessen im Umfeld serviceorientierter Architekturen*. ISBN 978-3-8364-1010-6, VDM Verlag Dr. Müller, 2007

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Abstrakte Dienstklassen zur Realisierung mobiler Prozesse*, In: TORSTEN BRAUN, GEORG CARLE und BURKHARD STILLER (Hrsg.) *Konferenzband zur KiVS 2007: Kommunikation in Verteilten Systemen - Band für Industriebeiträge, Kurzbeiträge und Workshops*, Seiten 123-128, VDE Verlag, 2007

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments*, In: GEORGE J. SUN (Hrsg.), *Journal of Computers*, ISSN 1796-203X, 2(1), Seite 1-11, Academy Publisher, 2007

CHRISTIAN P. KUNZE, SONJA ZAPLATA und WINFRIED LAMERSDORF, *Mobile Process Description and Execution*, In: FRANK ELIASSEN und ALBERTO MONTRESOR (Hrsg.), *Proceedings of the International Conference on Distributed Applications and Interoperable Systems (DAIS)*, Seiten 32-47, Springer, 2006

CHRISTIAN P. KUNZE *Unterstützung mobiler Prozesse im Mobile Computing,* In: FALKO DRESSLER und JÜRGEN KLEINÖDER (Hrsg.), *Technischer Bericht zum 1. GI/ITG KuVS Fachgespräch Energiebewusste Systeme und Methoden*, Seiten 42-47, Universität Erlangen-Nürnberg, 2005

CHRISTIAN P. KUNZE *DEMAC: A Distributed Environment for Mobility Aware Computing,* In: A. FERSCHA, R. MAYRHOFER, T. STRANG, C. LINNHOFF-POPIEN, A. DEY, A. BUTZ und SCHMIDT A. (Hrsg.), *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, Seiten 115–121, Österreichische Computer Gesellschaft, 2005.

CHRISTIAN P. KUNZE und TOBIAS BAIER *Identity Management for Self-Portrayal,* In: YVES DESWARTE, FRÉDÉRIC CUPPENS, SUSHIL JAJODIA und LINGYU WANG (Hrsg.), *Information Security Management, Education and Privacy*, Seiten 231–244. IFIP, Kluwer Academic Press, 2004.

CHRISTIAN P. KUNZE und TOBIAS BAIER *Identity-Enriched Session Management,* In: W. LAMERSDORF, V. TSCHAMMER und STÉPHANE AMARGER (Hrsg.), *Building the E-Service Society: E-Commerce, E-Business, and E-Government*, Seiten 329–342. IFIP, Kluwer Academic Publishers, 2004.

CHRISTIAN P. KUNZE *Digitale Identität und Identitäts-Management,* In: GESELLSCHAFT FÜR INFORMATIK E.V. (Hrsg.), *Informatiktage 2003*, Seiten 197–201, 2003.

Literaturverzeichnis

[AAGC04] ANTOLLINI, JOSE, MARIO ANTOLLINI, PABLO GUERRERO und MARIANO CILIA: *Extending Rebeca to Support Concept-Based Addressing*. In: *In Proceedings of the Argentinean Symposium on Information Systems (ASIS'04)*, Cordoba, Argentina, September 2004.

[AAH+97] ABOWD, GREGORY D., CHRISTOPHER G. ATKESON, JASON HONG, SUE LONG, ROB KOOPER und MIKE PINKERTON: *Cyberguide: a Mobile Context-Aware Tour Guide*. Wirel. Netw., 3(5):421–433, 1997.

[Aal00] AALST, WIL VAN DER: *Loosely Coupled Interorganizational Workflows: modeling and analyzing workflows crossing organizational boundaries*. Information and Management, 37(2):67–75, 2000.

[ACS06] ALEKSY, MARKUS, JAN CZERANSKI und MARTIN SCHADER: *Improving the Interoperability between Web Services and CORBA Using Pontifexv A Generic Bridge Generator*. In: *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, Seiten 166–171, 2006.

[Ada06] ADAM, HEIKO: *Konzeption und Integration eines Verzeichnisdienstes für mobile Systeme*. Diplomarbeit, Universität Hamburg, Department Informatik, Arbeitsbereich Verteilte Systeme und Informationssysteme, 2006.

[ADW01] ACKERMANN, MARK, TREVOR DARELL und DANIEL J. WEITZNER: *Privacy in Context*. HUMAN-COMPUTER INTERACTION, 16:167–176, 2001.

[AGRS05] ADELSTEIN, FRANK, SANDEEP K.S. GUPTA, GOLDEN G. RICHARD III und LOREN SCHWIEBERT: *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hill, 2005.

[AH02] AALST, WIL VAN DER und KEES VAN HEE: *Workflow Management - Models, Methods, and Systems*. The MIT Press, 2002.

[All05] ALLWEYER, THOMAS: *Geschäftsprozessmanagement : Strategie, Entwurf, Implementierung, Controlling*. W3L-Verlag, 2005.

[Bar05] BARDRAM, JAKOB E.: *The Java Context-Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications*. Lecture Notes in Computer Science (Proceedings of PERVASIVE 2005), 3468:98, 2005.

[BBB01] BENERECETTI, MASSIMO, PAOLO BOUQUET und MATTEO BONIFACIO: *Distributed Context-Aware Systems*. HUMAN-COMPUTER-INTERACTION, 16:213 – 228, 2001.

[BBE+07] BARRETO, CHARLTON, VAUGHN BULLARD, THOMAS ERL, JOHN EVDEMON, DIANE JORDAN, KHANDERAO KAND, DIETER KÖNIG, SIMON MOSER, RALPH STOUT, RON TEN-HOVE, IVANA TRICKOVIC, DANNY VAN DER RIJN und ALEX YIU: *Web Services Business Process Execution Language Version 2.0 - Primer*. Technischer Bericht, Organization for the Advancement of Structured Information Standards, 2007. Version vom 9. Mai 2007.

[BCS01] BELLAVISTA, PAOLO, ANTONIO CORRADI und CESARE STEFANELLI: *Mobile Agent Middleware for Mobile Computing*. IEEE Computer, 34(3):73–81, 2001.

[BDO06] BARROS, ALISTAIR R., MARLON DUMAS und PHILLIPA OAKS: *Standards for Web Service Choreography and Orchestration: Status and Perspectives*. In: *Business Process Management Workshops*, Band 3812/2006 der Reihe *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[Bec04] BECKER, CHRISTIAN: *System Support for Context-Aware Computing*. Habilitation, Universität Stuttgart, 2004.

[BFar05] B'FAR, REZA: *Mobile Computing Principles*. Cambridge University Press, 2005.

[BK04a] BAIER, TOBY und CHRISTIAN P. KUNZE: *Identity-Enriched Session Management*. In: LAMERSDORF, W., V. TSCHAMMER und STÉPHANE AMARGER (Herausgeber): *Building the E-Service Society: E-Commerce, E-Business, and E-Government*, Seiten 329–342. IFIP, Kluwer Academic Publishers Dordrecht, 8 2004.

[BK04b] BAIER, TOBY und CHRISTIAN P. KUNZE: *Identity Management for Self-Portrayal*. In: DESWARTE, YVES, FRÉDÉRIC CUPPENS, SUSHIL JAJODIA und LINGYU WANG (Herausgeber): *Information Security Management, Education and Privacy*, Seiten 231–244. IFIP, Kluwer Academic Press, 8 2004.

[BKW02] BAUS, JÖRG, ANTONIO KRÜGER und WOLFGANG WAHLSTER: *A Resource-Adaptive Mobile Navigation System*. In: *International Conference on Intelligent User Interfaces (IUI 02)*, Seiten 13–16, San Francisco, CA, USA, 2002. ACM Press.

[Blu04] BLUETOOTH QUALIFICATION REVIEW BOARD (BQRB): *Bluetooth Specification Version 2.0 + EDR*. Technischer Bericht, Bluetooth SIG, 2004.

[BN04] BECKER, CHRISTIAN und DANIELA NICKLAS: *Where do spacial context-models end and where do ontologies start? A proposal of a combined approach*. In: *Proceedings of Ubicomp 2004 workshop on Advanced Context Modeling, Reasoning and Management*, 2004.

[BST04] BELLAVISTA, PAOLO, CESARE STEFANELLI und MAURO TORTONESI: *The ubiQoS Middleware for Audio Streaming to Bluetooth Devices*. In: *Proceedings of Mobile and Ubiquitous Systems: Networking and Services, 2004 (MOBIQUITOUS)*, Seiten 138–145, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[Bun07] BUNDESVERBAND INFORMATIONSWIRTSCHAFT, TELEKOMMUNIKATION UND NEUE MEDIEN E. V.: *Daten zur Informationsgesellschaft – Status quo und Perspektiven Deutschlands im internationalen Vergleich*. Studie, BITKOM, 2007.

[BZL03] BAIER, TOBY, CHRISTIAN ZIRPINS und WINFRIED LAMERSDORF: *Digital Identity: How To Be Someone On*

The Net. In: REIS, ANTONIO PALMA DOS und PEDRO ISAIAS (Herausgeber): *e-Society 2003*, Seiten 815–820. IADIS International association for development of the information society, IADIS Press Lisbon, Portugal, 6 2003.

[CACM05] CHETAN, SHIVA, JALAL AL-MUHTADI, ROY CAMPBELL und M.DENNIS MICKUNAS: *Mobile Gaia: A Middleware for Adhoc Pervasive Computing*. In: *IEEE Consumer Communications & Networking Conference (CCNC 2005)*, Seiten 223–228. IEEE - Computer Society, 2005.

[Cap02] CAPRA, LICIA: *Mobile computing middleware for context-aware applications*. In: *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, Seiten 723–724, New York, NY, USA, 2002. ACM Press.

[CBAC05] CHAN, ELLICK, JIM BRESLER, JALAL AL-MUHTADI und ROY CAMPBELL: *Gaia Microserver: An Extendable Mobile Middleware Platform*. In: *Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05)*, Seiten 309–313. IEEE - Computer Society, 2005.

[CCC+06] CHALMERS, DAN, MATTHEW CHALMERS, JON CROWCROFT, MARTA KWIATKOWSKA, ROBIN MILNER, EAMONN O'NEILL, TOM RODDEN, VLADIMIRO SASSONE und MORRIS SLOMAN: *Ubiquitous Computing: Experience, Design and Science (Version 4, 23/2/06)*. Technischer Bericht, UKCRC Grand Challenges for Computing Research - Ubiquitous Computing Grand Challenge Section, 2006.

[CCF+05] CABRERA, LUIS FELIPE, GEORGE COPELAND, MAX FEINGOLD, ROBERT W. FREUND, TOM FREUND, SEAN JOYCE, JOHANNES KLEIN, DAVID LANGWORTHY, MARK LITTLE, FRANK LEYMANN, ERIC NEWCOMER, DAVID ORCHARD, IAN ROBINSON, TONY STOREY und SATISH THATTE: *Web Services Business Activity Framework (WS-BusinessActivity)*. Spezifikation August 2005, IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA, 2005.

[CCK+01] CLARK, JIM, CORY CASANAVE, KURT KANASKIE, BETTY HARVEY, JAMIE CLARK, NEAL SMITH, JOHN YUNKER und

KARSTEN RIEMER: *ebXML Business Process Specification Schema Version 1.01*. Spezifikation, 2001.

[CCL03] CHLAMTAC, IMRICH, MARCO CONTI und JENNIFER J.-N. LIU: *Mobile ad hoc networking: imperatives and challenges*. Ad Hoc Networks, 1:13–64, 2003.

[CCMW01] CHRISTENSEN, ERIK, FRANCISCO CURBERA, GREG MEREDITH und SANJIVA WEERAWARANA: *Web Services Description Language (WSDL) 1.1*. Spezifikation Note 15 March 2001, World Wide Web Consortium, 2001.

[CDK01] COULOURIS, GEORGE, JEAN DOLLIMORE und TIM KINDBERG: *Distributed Systems: Concepts and Design*. Addison-Wesley, 3 Auflage, 2001.

[CDMF00] CHEVERST, KEITH, NIGEL DAVIES, KEITH MITCHELL und ADRIAN FRIDAY: *Experiences of developing and deploying a context-aware tourist guide: the GUIDE project*. In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, Seiten 20–31, New York, NY, USA, 2000. ACM Press.

[CDN01] CUGOLA, GIANPAOLO und ELISABETTA DI NITTO: *Using publish/subscribe middleware for mobile systems*. In: *Proceedings of the Advanced Topic Workshop Middleware for Mobile Computing*, 2001.

[CEM01] CAPRA, LICIA, WOLFGANG EMMERICH und CECILIA MASCOLO: *Middleware for Mobile Computing: Awareness vs. Transparency*. In: *In Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, 2001. extended version.

[Cer02] CERAMI, ETHAN: *Web Services Essentials*. O'Reilly, 2002.

[CFJ03] CHEN, HARRY, TIM FININ und ANUPAM JOSHI: *An Intelligent Broker for Context-Aware Systems*. In: *Adjunct Proceedings of Ubicomp*, Seiten 183–184, 2003.

[CK00] CHEN, GUANLING und DAVID KOTZ: *A Survey of Context-Aware Mobile Computing Research*. Technischer Bericht TR2000-381, Department of Computer Science, Dartmouth College, 2000.

[CK02] CHEN, GUANLING und DAVID KOTZ: *Solar: An Open Platform for Context-Aware Mobile Applications (Short Paper)*. In: *Proceedings of the First International Conference on Pervasive Computing*, Seiten 41–47, 2002.

[Cla01] CLARK, DAVID: *Face-to-Face with Peer-to-Peer Networking*. Computer, 34(1):18–21, 2001.

[DA99] DEY, ANIND K. und GREGORY D. ABOWD: *Towards a Better Understanding of Context and Context-Awareness*. Technischer Bericht GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, 1999.

[DAH05] DUMAS, MARLON, WIL M. VAN DER AALST und ARTHUR H.M. TER HOFSTEDE: *Process-Aware Information Systems : Bridging People and Software through Process Technology*. John Wiley & Sons, 2005.

[Dav05] DAVIS, ROB: *Business process Modelling with ARIS – A Practical Guide*. Springer-Verlag, 2005.

[Dey00] DEY, ANIND K.: *Enabling the use of context in interactive applications*. In: *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, Seiten 79–80, New York, NY, USA, 2000. ACM Press.

[Dey01] DEY, ANIND K.: *Understanding and Using Context*. Personal and Ubiquitous Computing Journal, 5(1):4–7, 2001.

[DHN+04] DÜRR, FRANK, NICOLA HÖNLE, DANIELA NICKLAS, CHRISTIAN BECKER und KURT ROTHERMEL: *Nexus–A Platform for Context-Aware Applications*. In: ROTH, JÖRG (Herausgeber): *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, 2004.

[DJMZ05] DOSTAL, WOLFGANG, MARIO JECKLE, INGO MELZER und BABARA ZENGLER: *Service-orientierte Architekturen mit Web Services : Konzepte – Standards – Praxis*. Spektrum Akademischer Verlag, 2005.

[DO03] DOVAL, DIEGO und DONAL O'MAHONY: *Overlay Networks: A Scalable Alternative for P2P*. IEEE Internet Computing, 7(4):79–82, 2003.

[DPC+05] DIAZ, GREGORIO, JUAN-JOSÉ PARDO, MARÍA-EMILIA CAMBRONERO, VALENTÍN VALERO und FERNANDO CUARTERO: *Formal Techniques for Computer Systems and Business Processes*, Band 3670/2005 der Reihe *Lecture Notes in Computer Science*, Kapitel Automatic Translation of WS-CDL Choreographies to Timed Automata, Seiten 230–242. Springer-Verlag, 2005.

[DSFA99] DEY, ANIND K., DANIEL SALBER, MASAYASU FUTAKAWA und GREGORY D. ABOWD: *An Architecture to Support Context-Aware Applications*. Technischer Bericht GIT-GUV-99-23, College of Computing, Georgia Institute of Technology, 1999.

[Dub08] DUBRAY, JEAN-JACQUES: *WS-Choreography Definition Language (WS-CDL)*. Web-Seite, 2008. `http://www.ebpml.org/ws_-_cdl.htm`, Abruf am 02.01.2008.

[EFGK03] EUGSTER, PATRICK TH., PASCAL A. FELBER, RACHID GUERRAOUI und ANNE-MARIE KERMARREC: *The many faces of publish/subscribe*. ACM Computing Surveys, 35(2):114–131, 2003.

[EGB04] ESCHENAUER, LAURENT, VIRGIL D. GLIGOR und JOHN BARAS: *On Trust Establishment in Mobile Ad-Hoc Networks*. In: *Security Protocols*, Band 2845/2003 der Reihe *Lecture Notes in Computer Science*, Seiten 47 – 66. Springer-Verlag, 2004.

[Ell99] ELLIS, CLARENCE A.: *Workflow Technology*, Kapitel Computer Supported Co-operative Work, Seiten 29 – 54. John Wiley & Sons, 1999.

[EM88] ENGELMORE, ROBERT S. und ANTHONY J. MORGAN (Herausgeber): *Blackboard Systems*, Band III. Addison-Wesley, 1988.

[EN02] ELMASRI, RAMEZ und SHAMKANT B. NAVATHE: *Grundlagen von Datenbanksystemen*, Band 3. Pearson-Studium, 2002.

[FGKZ03] FIEGE, LUDGER, FELIX C. GÄRTNER, OLIVER KASTEN und ANDREAS ZEIDLER: *Supporting Mobility in Content-Based*

Publish/Subscribe Middleware. In: GOOS, G., J. HARTMANIS und J. VAN LEEUWEN (Herausgeber): *Proceedings of the ACM/IFIP/USENIX International Middleware Conference*, Band 2672/2003 der Reihe *Lecture Notes in Computer Science*, Seiten 103–122. Springer Berlin / Heidelberg, 2003.

[FHA99] FREEMAN, ERIC, SUSANNE HUPFER und KEN ARNOLD: *JavaSpaces$^{(TM)}$ Principles, Patterns, and Practice*. Pearson Education, 1999.

[FHKB05] FUCHS, FLORIAN, IRIS HOCHSTATTER, MICHAEL KRAUSE und MICHAEL BERGER: *A Meta-Model Approach to Context Information*. In: *Proceedings of 2nd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea) (at 3rd IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)), Hawaii, USA*, März 2005.

[FHLM01] FLOCH, JACQUELINE, SVEIN HALLSTEINSEN, ARNE LIE und HANS I. MYRHAUG: *A Reference Model for Context-Aware Mobile Services*. In: *Proceedings of the Norsk Informatikkonferanse*, 2001.

[Fos05] FOSTER, IAN: *Service-oriented Science*. Science Magazine: All for One and One for All, 308(5723):814–817, 2005.

[Fre06] FREDLUND, LARS-ÅKE: *Implementing WS-CDL*. In: *Proceedings of JSWEB 2006 (II Jornadas Científico-Técnicas en Servicios Web)*, 2006.

[FWDB97] FRIDAY, ADRIAN, STEPHEN P. WADE, NIGEL DAVIES und GORDON S. BLAIR: *The Tuple Space: An Old Solution to a New Problem?* Technischer Bericht, Lancaster University, Computing Department, 1997.

[GB03] GRACE, PAUL und SAM BLAIR, GORDON S.AND SAMUEL: *Middleware Awareness in Mobile Computing*. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Seiten 382–387, 2003.

[GGKS02] GOTTSCHALK, KARL, STEPHEN GRAHAM, HEATHER KREGER und JAMES SNELL: *Introduction to Web services archi-*

tecture. IBM SYSTEMS JOURNAL: New Developments in Web Services and E-commerce, 41(2):170–177, 2002.

[GHJV03] GAMMA, ERICH, RICHARD HELM, RALPH JOHNSON und JOHN VLISSIDES: *Entwurfsmuster : Elemente wiederverwendbarer objektorientierter Software*. Professionelle Softwareentwicklung – Programmer's choice. Addison-Wesley, 5 Auflage, 2003.

[Gün03] GÜNTZEL, KAI: *Web Services-Based Transactional Workflows - Advanced Transaction Concepts*. In: *On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*, Band 2889/2003 der Reihe *Lecture Notes in Computer Science*, Seiten 70–82. Springer-Verlag, 2003.

[HA03] HOPKINS, BRUCE und RANJITH ANTONY: *Bluetooth for Java*. Springer-Verlag, 2003.

[Ham05] HAMMERSCHALL, ULRIKE: *Verteilte Systeme und Anwendungen - Architekturkonzepte, Standards und Middleware-Technologien*. Pearson Studium, 2005.

[HCE+06] HALL, JANE, ALEXANDRU CICORTAS, KLAUS-PETER ECKERT, FLORIN FORTIS, BENOIT GAILLARD, YURI GLICKMAN, RAFAL KNAPIK und RAFAL RENK: *D2.1 - VISP Workflow Technologies Functional Analysis and Comparison*. White Paper, Fraunhofer Institute for Open Communication Systems FOKUS, 2006.

[HDVL03] HELAL, SUMI, NITIN DESAI, VARUN VERMA und CHOONHWA LEE: *Konark – A Service Discovery and Delivery Protocol for Ad-Hoc Networks*. In: *Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC)*, 2003.

[HGM04] HUANG, YONGQIANG und HECTOR GARCIA-MOLINA: *Publish/Subscribe in a Mobile Environment*. Wireless Networks, 10(6):643–652, 2004.

[HL04] HONG, JASON I. und JAMES A. LANDAY: *An architecture for privacy-sensitive ubiquitous computing*. In: *MobiSys '04:*

Proceedings of the 2nd international conference on Mobile systems, applications, and services, Seiten 177–189, New York, NY, USA, 2004. ACM Press.

[HMNS03] HANSMANN, UWE, LOTHAR MERK, MARTIN S. NICKLOUS und THOMAS STOBER: *Pervasive Computing - The Mobile World*. Springer-Verlag, 2. Auflage, 2003.

[Hol04] HOLLINGSWORTH, DAVID: *The Workflow Reference Model 10 Years On*. In: FISCHER, LAYNA (Herausgeber): *Workflow Handbook 2004*, Seiten 295–312. Lighthouse Point, 2004.

[Hol07] HOLBREICH, ALEXANDER: *Transaktionsunterstützung für verteilt ausgeführte Mobile Prozesse*. Diplomarbeit, Universität Hamburg, 2007.

[HSP+03] HOFER, THOMAS, WIELAND SCHWINGER, MARIO PICHLER, GERHARD LEONHARTSBERGER, JOSEF ALTMANN und WERNER RETSCHITZEGGER: *Context-Awareness on Mobile Devices - the Hydrogen Approach*. In: *Proceedings of the 36th Hawaii International Conference on System Sciences*, Seiten 292–302. IEEE - Computer Society, 2003.

[HTKR05] HÖPFNER, HAGEN, CAN TÜRKER und BIRGITTA KÖNIG-RIES (Herausgeber): *Mobile Datenbanken und Informationssysteme - Konzepte und Techniken*. dpunkt-Verlag, 2005.

[HW05] HRASTNIK, PETER und WERNER WINIWARTER: *TWSO - TransactionalWeb Service Orchestrations*. In: *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05)*, Seiten 45–50. IEEE Computer Society, 2005.

[HZJZ04] HU, JIANQIANG, BIN ZHOU, YAN JIA und PENG ZOU: *Research and Implementation of CORBA Web Services*. In: *Conceptual Modeling for Advanced Application Domains*, Band 3289/2004 der Reihe *Lecture Notes in Computer Science*, Seiten 314–322. Springer-Verlag, 2004.

[IEE99] IEEE COMPUTER SOCIETY: *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control*

(MAC) and Physical Layer (PHY) Specifications. Spezifikation ANSI/IEEE Std 802.11, 1999 Edition (R2003), IEEE Computer Society, 1999.

[II05] ISO und IEC: *ISO/IEC 9834-8: Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*. Internationaler Standard ISO/IEC 9834-8:2005(E), International Organization for Standardization / International Electrotechnical Commission, 2005.

[IM05] ILYAS, MOHAMMAD und IMAD MAHGOUB (Herausgeber): *Mobile Computing Handbook*. Auerbach Publications, 2005.

[Int03] INTEL CORPORATION: *Understanding Wi-Fi and WiMAX as Metro-Access Solutions*. Technischer Bericht, Intel Corporation, 2003.

[Jab97] JABLONSKI, STEFAN: *Architektur von Workflow-Management-Systemen*. Informatik - Forschung und Entwicklung, 12(7):72–81, 1997.

[Jab98] JABLONSKI, STEFAN: *A Software Architecture for Workflow Management Systems*. In: *Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA'98)*, Seiten 739 – 744. IEEE - Computer Society, 1998.

[JBo07] JBOSS INC.: *JBoss jBPM jPDL 3.2 – jBPM jPDL User Guide*. Webseite (Dokumentation), 2007. `http://docs.jboss.com/jbpm/v3/userguide/index.html`, Abruf am 17.08.2007.

[Jos06] JOSEFSSON, SIMON: *The Base16, Base32, and Base64 Data Encodings*. Internet-Standard RFC 4648, The Internet Society – Network Working Group, 2006.

[JPK99] JUNG, DO-GUEN, KWANG-JIN PAEK und TAI-YUN KIM: *Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing*. In: *ICPP '99: Proceedings of the*

1999 International Conference Workshops on Parallel Processing, Seite 434, Washington, DC, USA, 1999. IEEE Computer Society.

[KHA03] KIEPUSZEWSKI, BARTEK, ARTHUR H. M. TER HOFSTEDE und WIL M. VAN DER AALST: *Fundamentals of Control Flow in Workflows*. Acta Informatica, 39(3):143 – 209, 2003.

[KBR+05] KAVANTZAS, NICKOLAS, DAVID BURDETT, GREGORY RITZINGER, TONY FLETCHER, YVES LAFON und CHARLTON BARRETO: *Web Services Choreography Description Language Version 1.0*. Spezifikation CR-ws-cdl-10-20051109, World Wide Web Consortium, 2005. http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/, Abruf am 23.08.2007.

[KHW+02] KUTVONEN, LEA, JENNY HUANG, ANDREA WESTERINEN, MICHELLE SIBILLA, THIERRY DESPRATS, ANDRÁ BARROS DE SALES und DOMINIQUE BENECH: *Middleware Management*. White Paper, OMG Telecom Domain Task Force, 2002.

[KL04] KOSSMANN, DONALD und FRANK LEYMANN: *Web Services*. Informatik Spektrum, 27(2):117–128, 2004.

[Kle96] KLEINROCK, LEONARD: *Nomadic Computing*. In: *Information Network and Data Communication*, Seiten 223–233, 1996.

[KMK+03] KORPIPÄÄ, PANU, JANI MÄNTYJÄRVI, JUHA KELA, HEIKKI KERÄNEN und ESKO-JUHANI MALM: *Managing Context Information in Mobile Devices*. IEEE Pervasive Computing, 02(3):42–51, 2003.

[KMS+05] KORPIPÄÄ, PANU, ESKO-JUHANI MALM, ILKKA SALMINEN, TAPANI RANTAKOKKO, VESA KYLLÖNEN und ILKKA KÄNSÄLÄ: *Context Management for End user Development of Context-Aware Applications*. In: *MDM '05: Proceedings of the 6th international conference on Mobile data management*, Seiten 304–308, New York, NY, USA, 2005. ACM Press.

[Kön07] KÖNIG, DIETER: *WS-BPEL 2.0 – Language Constructs Overview*. Online-Präsentation, 2007. http://www.oasis-open.org/committees/download.php/

23067/WS-BPEL␣TechnicalOverview␣forDevelopers␣andArchitects␣-␣Part2␣(DieterKoenig).pdf, Abruf am 30.05.2007.

[Kor05] KORPIPÄÄ, PANU: *Blackboard-based software framework and tool for mobile device context awareness*. Doktorarbeit, Electronics in conjunction with University of Oulu, 2005.

[KP03] KADDOUR, MEJDI und LAURENT PAUTET: *Towards an Adaptable Message Oriented Middleware for Mobile Environments*. In: *Proceedings of the IEEE 3rd workshop on Applications and Services in Wireless Networks*, Berne, Switzerland, 2003.

[Kun05a] KUNZE, CHRISTIAN P.: *DEMAC: A Distributed Environment for Mobility Aware Computing*. In: FERSCHA, A., R. MAYRHOFER, T. STRANG, C. LINNHOFF-POPIEN, A. DEY, A. BUTZ und SCHMIDT A. (Herausgeber): *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, Seiten 115–121. Oesterreichische Computer Gesellschaft, 5 2005.

[Kun05b] KUNZE, CHRISTIAN P.: *Unterstützung mobiler Prozesse im Mobile Computing*. In: DRESSLER, FALKO und JÜRGEN KLEINÖDER (Herausgeber): *Techischer Bericht zum 1. GI/ITG KuVS Fachgespräch Energiebewusste Systeme und Methoden*, Seiten 42–47. Friedrich-Alexander-Universiät Erlangen-Nürnberg, Technische Fakultät, Institut für Informatik, 2005.

[KZL06] KUNZE, CHRISTIAN P., SONJA ZAPLATA und WINFRIED LAMERSDORF: *Mobile Process Description and Execution*. In: ELIASSEN, FRANK und ALBERTO MONTRESOR (Herausgeber): *Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, Seiten 32 – 47. Springer, 2006.

[KZL07a] KUNZE, CHRISTIAN P., SONJA ZAPLATA und WINFRIED LAMERSDORF: *Abstrakte Dienstklassen zur Realisierung mobiler Prozesse*. In: *Konferenzband zur Konferenz: Kommunikation in Verteilten Systemen (KiVS) 2007*, Seiten 123–128. Gesellschaft für Informatik, 2007.

[KZL07b] KUNZE, CHRISTIAN P., SONJA ZAPLATA und WINFRIED LAMERSDORF: *Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments*. Journal of Computers, 2(1):1–11, 2007.

[KZTL08] KUNZE, CHRISTIAN P., SONJA ZAPLATA, MIRWAIS TURJALEI und WINFRIED LAMERSDORF: *Enabling Context-based Cooperation: A Generic Context Model and Management System*. In: *Proceedings of the 11th International Conference on Business Information Systems (BIS 2008)*, Lecture Notes in Business Information Processing (LNBIP). Springer-Verlag, 2008.

[Lak03] LAKKALA, HARRI: *Context Exchange Protocol Specification: Version 1.0*. Spezifikation, Nokia Corporation, 2003.

[Lam94] LAMERSDORF, WINFRIED: *Datenbanken in verteilten Systemen - Konzepte, Lösungen, Standards*. Vieweg, 1994.

[LBBN04] LEHMANN, OTHMAR, MARTIN BAUER, CHRISTIAN BECKER und DANIELA NICKLAS: *From Home to World - Supporting Context-aware Applications through World Models*. Percom, Seiten 297–306, 2004.

[Ley03] LEYMANN, FRANK: *Choreography: Geschäftsprozesse mit Web-Services*. OBJECTspektrum, 6:13–16, 2003.

[LH02] LEE, CHOONHWA und SUMI HELAL: *Protocols for Service Discovery in Dynamic and Mobile Networks*. International Journal of Computer Research, 11(1):1–12, 2002.

[LNW08] LEYMANN, FRANK, DANIELA NICKLAS und MATTHIAS WIELAND: *SFB 627: Nexus – Teilprojekt E1: Kontextbezogene Workflows*. Web-Seite, 2008. `http://www.iaas.uni-stuttgart.de/forschung/projects/nexus/fp2/tpe1.html`, Abruf am 01.01.2008.

[LR00] LEYMANN, FRANK und DIETER ROLLER: *Production Workflow - Concepts and Techniques*. Prentice Hall, 2000.

[MA99] MARIN, MIKE und MIKE ANDERSON: *Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification*. Spezifikation WFMC-TC-1012, Workflow Management Coalition, 1999.

[Mah01] MAHMOUD, QUSAY H.: *Distributed Programming with Java*. Manning Publications, 2001.

[Mat01] MATTERN, FRIEDEMANN: *Pervasive / Ubiquitous Computing*. Informatik-Spektrum, 24(3):145–147, 2001.

[Mat03] MATTERN, FRIEDEMANN: *Vom Verschwinden des Computers – Die Vision des Ubiquitous Computing*. In: MATTERN, FRIEDEMANN (Herausgeber): *Total vernetzt*, Seiten 1–41. Springer-Verlag, 2003.

[Mat05a] MATTERN, FRIEDEMANN: *Acht Thesen zur Informatisierung des Alltags*. In: *Tagungsband zum Acatech-Symposium Computer in der Alltagswelt*, Berlin, Juni 2005.

[Mat05b] MATTERN, FRIEDEMANN: *Allgegenwärtige und verschwindende Computer*. Praxis der Informationsverarbeitung und Kommunikation (PIK), 28(1):29–36, 2005.

[Mat06] MATTERN, FRIEDEMANN: *Allgegenwärtige Informationsverarbeitung – Technologietrends und Auswirkungen des Ubiquitous Computing*. Internetveröffentlichung, 2006. *http://www.vs.inf.ethz.ch/publ/papers/AllgegenwInfoverarb.pdf*, Abruf am 26.09.2006.

[MCE02] MASCOLO, CECILIA, LICIA CAPRA und WOLFGANG EMMERICH: *Middleware for Mobile Computing (A Survey)*. In: GREGORI, E., G. ANASTASI und S. BASAGNI (Herausgeber): *Neworking 2002 Tutorial Papers*, Band 2497 der Reihe *Lecture Notes in Computer Science*, Seiten 20–58. Springer-Verlag, 2002.

[Mel04] MELNIK, SERGEY: *Generic Model Management: Concepts and Algorithms*. Springer, 2004.

[Mil06] MILNER, ROBIN: *Ubiquitous Computing: Shall we Understand It?* The Computer Journal, 49(4):383–389, 2006.

[MN02] MARIN, MIKE und ROBERTA NORIN: *Workflow Process Definition Interface – XML Process Definition Language*. Specification WFMC-TC-1025, Workflow Management Coalition, 2002.

[MPR06] MURPHY, AMY L., GIAN PIETRO PICCO und GRUIA-CATALIN ROMAN: *LIME: A coordination model and middleware supporting mobility of hosts and agents*. ACM Transactions on Software Engineering and Methodology (TOSEM), 15(3):279–328, 2006.

[MTK99] MATSUI, SUSUMU, YASUHIRO TAHAHASHI und TAKESHI KONDO: *Development of Communication Software for Mobile Computing*. Hitachi Review, 48:246–250, 1999.

[NGS+01] NICKLAS, DANIELA, MATTHIAS GROSSMANN, THOMAS SCHWARZ, STEFFEN VOLZ und BERNHARD MITSCHANG: *A Model-Based, Open Architecture for Mobile, Spatially Aware Applications*. In: C.S. JENSEN ET AL. (Herausgeber): *Advances in Spatial and Temporal Databases*, Band 2121 der Reihe *Lecture Notes in Computer Science*, Seiten 117–135. Springer-Verlag, 2001.

[OAS04] OASIS: *Introduction to UDDI: Important Features and Functional Concepts*. Technical White Paper, Organization for the Advancement of Structured Information Standards, 2004.

[Obj06] OBJECT MANAGEMENT GROUP: *CORBA to WSDL/SOAP Interworking Specification Version 1.2*. Spezifikation formal/06-11-01, Object Management Group, 2006.

[OEt02] O'SULLIVAN, JUSTIN, DAVID EDMOND und ARTHUR H.M. TER HOFSTEDE: *What's in a Service?* Distributed and Parallel Databases, 12(2):117–133, 2002.

[OMG99] OMG: *The Common Object Request Broker: Architecture and Specification*. Technischer Bericht V 2.3, Object Management Group, 1999.

[Ora01] ORAM, ANDY: *Peer-to-Peer – Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[Pap03] PAPAZOGLOU, MIKE P.: *Service-Oriented Computing: Concepts, Characteristics and Directions*. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, Seiten 3–12. IEEE - Computer Society, 2003.

[Pas97] PASCOE, JASON: *The stick-e note architecture: extending the interface beyond the user*. In: *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, Seiten 261–264, New York, NY, USA, 1997. ACM Press.

[PD04] PAPAZOGLOU, MICHAEL P. und JEAN-JACQUES DUBRAY: *A Survey of Web service technologies*. Technischer Bericht DIT-04-058, University of Trento, 2004.

[Pel03a] PELTZ, CHRIS: *Web Service Orchestration and Choreography*. Web Services Journal, 3, 2003.

[Pel03b] PELTZ, CHRIS: *Web Services Orchestration and Choreography*. Computer, 36(10):46–52, 2003.

[Per06] PERNICI, BARBARA: *Basic Concepts*. In: PERNICI, BARBARA (Herausgeber): *Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility*. Springer-Verlag, 2006.

[PG03] PAPAZOGLOU, MIKE P. und DIMITRIOS GEORGAKOPOULOS: *Service-oriented Computing*. Communications of the ACM, 46(10):24–28, 2003.

[PMR99] PICCO, GIAN PIETRO, AMY L. MURPHY und GRUIA-CATALIN ROMAN: *LIME: Linda Meets Mobility*. In: *ICSE '99: Proceedings of the 21st international conference on Software engineering*, Seiten 368–377, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[Pop98] POPE, ALAN: *The CORBA reference Guide - Understanding the Common Object Request Broker Architecture*. Addison-Wesley, 1998.

[PVZ+06] PORTILLA, ALBERTO, GENOVEVA VARGAS-SOLAR, JOSÉ-LUIS ZECHINELLI-MARTINI, CHRISTINE COLLET und LUCIANO GARCIA-BANUELOS: *A Survey for Analyzing Transactional Behavior in Service-based Applications*. In: *Proceedings of the Seventh Mexican International Conference on Computer Science (ENC'06)*, Seiten 116–126. IEEE - Computer Society, 2006.

[RBB03] ROTHERMEL, KURT, MARTIN BAUER und CHRISTIAN BECKER: *Digitale Weltmodelle - Grundlage kontextbezogener*

Systeme. In: MATTERN, FRIEDEMANN (Herausgeber): *Total vernetzt - Szenarien einer informatisierten Welt*, Kapitel 5, Seiten 123–141. Springer-Verlag, 2003.

[RCRM02] RANGANATHAN, ANAND, ROY H. CAMPBELL, ARATHI RAVI und ANUPAMA MAHAJAN: *ConChat: A Context-Aware Chat Program*. IEEE Pervasive Computing, 1(3):52–58, 2002.

[Rei03] REIJERS, HAJO A.: *Design and Control of Workflow Processes: Business Process Management for the Service Industry*. Nummer 2617 in *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[RFM+03] ROTHERMEL, KURT, DIETER FRITSCH, BERNHARD MITSCHANG, PAUL J. KÜHN, MARTIN BAUER, CHRISTIAN BECKER, CHRISTIAN HAUSER, DANIELA NICKLAS und STEFFEN VOLZ: *SFB 627 Umgebungsmodelle für mobile kontextbezogene Systeme*. In: DITTRICH, KLAUS, WOLFGANG KÖNIG, ANDREAS OBERWEIS, KAI RANNENBERG und WOLFGANG WAHLSTER (Herausgeber): *INFORMATIK 2003 - Innovative Informatikanwendungen*, Band 34 (1) der Reihe *Lecture Notes in Informatics*, Seiten 103–115. Gesellschaft für Informatik e.V., 2003.

[RHC+02a] ROMÁN, MANUEL, CHRISTOPHER K. HESS, RENATO CERQUEIRA, ANAND RANGANATHAN, ROY H. CAMPBELL und KLARA NAHRSTEDT: *Gaia: A Middleware Infrastructure to Enable Active Spaces*. IEEE Pervasive Computing, 1:74–83, 2002.

[RHC+02b] ROMÁN, MANUEL, CHRISTOPHER K. HESS, RENATO CERQUEIRA, ANAND RANGANATHAN, ROY H. CAMPBELL und KLARA NAHRSTEDT: *Gaia: A Middleware Platform for Active Spaces*. ACM SIGMOBILE: Mobile Computing and Communications Review, 6(4):65–67, 2002.

[Rie05] RIEGEN, MICHAEL VON: *Transaktionale Koordination dynamischer Prozesse in Grid-Umgebungen*. Diplomarbeit, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, 10 2005.

[RKM02] RÖMER, KAY, OLIVER KASTEN und FRIEDEMANN MATTERN: *Middleware Challenges for Wireless Sensor Networks*.

ACM SIGMOBILE: Mobile Computing and Communications Review, 6(4):59 – 61, 2002.

[Rot02a] ROTH, JÖRG: *Mobile Computing - Grundlagen, Technik, Konzepte*. dpunkt-Verlag, 2002.

[Rot02b] ROTH, JÖRG: *Patterns of Mobile Interaction*. Personal Ubiquitous Computing, 6(4):282–289, 2002.

[RPM00] ROMAN, GRUIA-CATALIN, GIAN PIETRO PICCO und AMY L. MURPHY: *Software engineering for mobility: a roadmap*. In: *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, Seiten 241–258. ACM Press, 2000.

[RR02] RAMANATHAN, RAM und JASON REDI: *A brief overview of ad hoc networks: challenges and directions*. IEEE - Communications Magazine, 40(5):20–22, 2002.

[SAS01] SUTTON, PETER, RHYS ARKINS und BILL SEGALL: *Supporting Disconnectedness-Transparent Information Delivery for Mobile and Invisible Computing*. In: *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, Seite 277, Washington, DC, USA, 2001. IEEE Computer Society.

[Sat96] SATYANARAYANAN, MAHADEV: *Fundamental Challenges in Mobile Computing*. In: *Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing*, 1996.

[SAT+99] SCHMIDT, ALBRECHT, KOFI ASANTE AIDOO, ANTTI TAKALUOMAI, URPO TUOMELAI, KRISTOF VAN LAERHOVEN und WALTER VAN DE VELDE: *Advanced Interaction in Context*. In: GELLERSEN, HANS-W. (Herausgeber): *Handheld and Ubiquitous Computing*, Nummer 1707 in *Lecture Notes in Computer Science*, Seiten 89–101. Springer Verlag, 1999.

[Sat03] SATYANARAYANAN, MAHADEV: *Privacy: The Achilles Heel of Pervasive Computing?* IEEE Pervasive Computing, 2(1):2–3, 2003.

[Sat04] SATYANARAYANAN, MAHADEV: *The Many Faces of Adaption*. IEEE Pervastive Computing, 3(3):4–5, 2004.

[SBG98] SCHMIDT, ALBRECHT, MICHAEL BEIGL und HANS-W. GELLERSEN: *There is more to Context than Location*. In: *Proceedings of the International Workshop on Interactive Applications of Mobile Computing*, 1998.

[Sch03] SCHILLER, JOCHEN: *Mobilkommunikation*. Pearson Studium, 2003.

[Sch05] SCHOCH, THOMAS: *Middleware für Ubiquitous-Computing-Anwendungen*. In: FLEISCH, ELGAR und FRIEDEMANN MATTERN (Herausgeber): *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer-Verlag, 2005.

[SGBH00] SIEP, THOMAS M., IAN C. GIFFORD, RICHARD C. BRALEY und ROBERT F. HEILE: *Paving the Way for Personal Area Network Standards: An Overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks*. IEEE Personal Communications, 7:37–43, 2000.

[Sin04] SINGHERA, ZAFAR U.: *Extended Web Services Framework to Meet Non-Functional Requirements*. In: *Proceedings of the 2004 International Symposium on Applications and the Internet Workshops (SAINTW'04)*, Seiten 334 – 340. IEEE - Computer Society, 2004.

[SLP04] STRANG, THOMAS und CLAUDIA LINNHOFF-POPIEN: *A Context Modeling Survey*. In: *Proceedings of the Workshop on Advanced Context Modeling, Reasoning and Management associated with the Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*, 2004.

[SMNT99] SAITO, TARO, YAHEI MURAKATA, MASATOSHI NAKAYA und NAOKO TANIGUCHI: *Middleware and Applications for Mobile Computers*. Hitachi Review, 48(1):21–26, 1999.

[SO04] SCHULZ, KARSTEN A. und MARIA E. ORLOWSKA: *Facilitating cross-organisational workflows with a workflow view approach*. Data & Knowledge Engineering, 51(1):109–147, 2004.

[SS02] SUN, JUN-ZHAO und JAAKKO SAUVOLA: *On fundamental concept of mobility for mobile communications*. Proceedings

of The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2:799– 803, 2002.

[SSJS03] SANG HYUN, PARK, WON SO HEE, LEE JONG BONG und KIM SUNG WOO: *Smart home – digitally engineered domestic life*. Personal and Ubiquitous Computing, 7(3):189 – 196, 2003.

[STM00] SCHMIDT, ALBRECHT, ANTTI TAKALUOMA und JANI MÄNTYJÄRVI: *Context-Aware Telephony Over WAP*. Personal Ubiquitous Comput., 4(4):225–229, 2000.

[Sun01] SUN MICROSYSTEMS: *Jini Network Technology – An Executive Overview*. White Paper, Februar 2001.

[SW05] STEINMETZ, RALF und KLAUS WEHRLE: *What Is This "Peer-to-Peer" About?* In: *Peer-to-Peer Systems and Applications*, Band 3485/2005 der Reihe *Lecture Notes in Computer Science*, Seiten 9–16. Springer-Verlag, 2005.

[Tan97] TANENBAUM, ANDREW S.: *Computernetzwerke*. Prentice Hall, 3 Auflage, 97.

[TP02] TSALGATIDOU, APHRODITE und THOMI PILIOURA: *An Overview of Standards and Related Technology in Web Services*. Distributed and Parallel Databases, 12(2-3):135–162, 2002.

[TS02] TANENBAUM, ANDREW S. und MAARTEN VAN STEEN (Herausgeber): *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[Tur06] TURJALEI, MIRWAIS: *Integration von Context-Awareness in eine Middleware für mobile Systeme*. Diplomarbeit, Universität Hamburg, Department Informatik, Verteilte Systeme und Informationssysteme, Juni 2006.

[VC99] VALIÑO, MARCOS und JUAN CORCHADO: *VoIP: The Convergence of Networks*. Computing and Information Systems, 6:105–112, 1999.

[WCC04] WELLS, GEORGE, ALAN CHALMERS und PETER G. CLAYTON: *Linda implementations in Java for concurrent systems*. Concurrency and Computation: Practice and Experience, 16(10):1005–1022, 2004.

[Wei91] WEISER, MARK: *The Computer for the Twenty-First Century*. Scientific American, 256(3):94–104, 1991.

[Wei93] WEISER, MARK: *Hot Topics: Ubiquitous Computing*. IEEE Computer, 1993.

[WGZP04] WANG, XIAO HANG, TAO GU, DA QING ZHANG und HUNG KENG PUNG: *Ontology Based Context Modeling and Reasoning using OWL*. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, Seiten 18–22. IEEE - Computer Society, 2004.

[Wil02] WILSON, BRENDON J.: *JXTA*. New Riders Publishing, 1 Auflage, 2002.

[WiM06] WIMAX FORUM: *Mobile WiMAX – Part I: A Technical Overview and Performance Evaluation*. Technischer Bericht, WiMAX Forum, 2006.

[Win01] WINOGRAD, TERRY: *Architectures for Context*. Human-Computer Intercation, 16:401–419, 2001.

[Win07] WINNICKI, ALICE: *Erweiterung einer Middleware für das Mobile Computing um nicht-funktionale Sicherheits- und Vertrauensaspekte*. Diplomarbeit, Universität Hamburg, Department Informatik, Verteilte Systeme und Informationssysteme, 2007.

[Wis05] WISNIEWSKI, STEVE: *Wireless and Cellular Networks*. Prentice Hall, 2005.

[WKNL07] WIELAND, MATTHIAS, OLIVER KOPP, DANIELA NICKLAS und FRANK LEYMANN: *Towards Context-Aware Workflows*. In: PERNICI, BARBARA und JON ATLE GULLA (Herausgeber): *CAiSE´07 Proceedings of the Workshops and Doctoral Consortium Vol.2, Trondheim, Norway, June 11-15th, 2007*. Tapir Academic Press, 2007.

[Wor99] WORKFLOW MANAGEMENT COALITION: *Terminology & Glossary*. Spezifikation WFMC-TC-1011, Workflow Management Coalition, 1999.

[WY05] WU, RUEY-SHYANG und SHYANG-MING YUAN: *Enable Transaction in Web Services Environment*. In: *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05)*, Seiten 15–20. IEEE Computer Society, 2005.

[YLDW05] YAN, SHIFENG, YING LI, SHUIGUANG DENG und ZHAOHUI WU: *A Transaction Management Framework for Service-Based Workflow*. In: *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05)*, Seiten 377–381. IEEE Computer Society, 2005.

[ZF03] ZEIDLER, ANDREAS und LUDGER FIEGE: *Mobility support with REBECA*. In: *Proceedings of the Workshop on Mobile Computing Middleware*, Seiten 354– 360, 2003.

[Zie05] ZIEGLE, CAI-NICOLASR: *Towards Decentralized Recommender Systems*. Dissertation, Albert-Ludwigs-Universität Freiburg, 2005.

[ZK07] ZAPLATA, SONJA und CHRISTIAN P. KUNZE: *Prozessmanagement im Mobile Computing - Kooperative Ausführung von Geschäftsprozessen im Umfeld serviceorientierter Architekturen*. VDM Verlag Dr. Müller, 2007.

Abkürzungsverzeichnis

2PC	Zwei-Phasen-Commit-Protokoll
AES	Advanced Encryption Standard
Akku	Akkumulator
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat-Request
AWML	Augmented World Modeling Language
AWQL	Augmented World Query Language
BNEP	Bluetooth Network Encapsulation Protocol
BPEL4WS	Business Process Execution Language for Web Services
BSS	Basic Service Set
CDR	Common Data Representation
CEP	Context Exchange Protocol
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
DEMAC	Distributed Environment for Mobility-Aware Computing
DFG	Deutsche Forschungsgemeinschaft
DPDL	DEMAC Process Description Language
DS	Distributionssystem
DSL	Digital Subscriber Line
EbBPSS	EbXML Business Process Specification Schema
ebXML	Electronic Business Using XML
ECA	Event-Condition-Action
EIA	Electronic Industries Alliance
ESS	Extended Service Set
FCFS	First Come – First Served
FIFO	First In – First Out
FTP	File Transfer Protocol
GIOP	General Inter-ORB Protocol
GSM	Global System for Mobile Communications
HCI	Host Controller Interface
HID	Human Interface Device (Protocol)
HTML	Hypertext Markup Language

HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IBSS	Independent Basic Service Set
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
IP	Internet Protocol
IrDA	Infrared Data Association
ISO	International Organization for Standardization
IuK	Informations- und Kommunikationstechnologie
jBPM	Java Business Process Management
JPDL	jBPM Process Definition Language
L2CAP	Logical Link Control and Adaption Protocol
LAN	Local Area Network
LKW	Lastkraftwagen
LLC	Logical Link Control
MAC	Media Access Control
MBit	Mega-Bit
MHz	Mega-Hertz
MOM	Message-Oriented Middleware
MVC	Model View Controller
OASIS	Organization for the Advancement of Structured Information Standards
OBEX	Object Exchange
OLTP	Online Transaction Processing
ORB	Object Request Broker
OSI	Open System Interconnection
P2P	Peer-to-Peer
PC	Personal Computer
PDA	Persönlicher Digitaler Assistent (Personal Digital Assistant)
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
POS	Personal Operating Space
PPP	Point-to-Point Protocol
QoS	Quality of Service
RDP	Reliable Datagram Protocol

RFC	Request for Comments
RFCOMM	Radio Frequency Communication
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
S-OFDM	Scalable Orthogonal Frequency Division Multiplexing
SDP	Service Discovery Protocol
SHA	Secure Hash Algorithm
SOA	Service-Oriented Architecture
TCP	Transport Control Protocol
TCS	Telephony Control Protocol Specification
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
vgl.	vergleiche
VIT	Virtual Information Tower
VoIP	Voice over Internet Protocol
W3C	World Wide Web Consortium
WAP	Wireless Access Protocol
WfMC	Workflow Management Coalition
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WPAN	Wireless Personal Area Network
WS-BPEL	Web Service Business Process Execution Language
WSCDL	Web Services Choreography Description Language
WSCI	Web Service Choreographie Interface
WSDL	Web Service Definition Language
WWAN	Wireless Wide Area Network
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XPDL	XML Process Definition Language
z. T.	zum Teil

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Hamburg, den 26. Juni 2008

Christian P. Kunze