Jörg Lonthoff

Externes Anwendungsmanagement

GABLER EDITION WISSENSCHAFT

Information Engineering und IV-Controlling

Die Schriftenreihe präsentiert aktuelle Forschungsergebnisse der Wirtschaftsinformatik sowie interdisziplinäre Ansätze aus Informatik und Betriebswirtschaftslehre. Ein zentrales Anliegen ist dabei die Pflege der Verbindung zwischen Theorie und Praxis durch eine anwendungsorientierte Darstellung sowie durch die Aktualität der Beiträge. Mit der inhaltlichen Orientierung an Fragen des Information Engineerings und des IV-Controllings soll insbesondere ein Beitrag zur theoretischen Fundierung und Weiterentwicklung eines wichtigen Teilbereichs der Wirtschaftsinformatik geleistet werden.

Jörg Lonthoff

Externes Anwendungsmanagement

Organisation des Lebenszyklus komponentenbasierter, mobiler Anwendungen

Mit einem Geleitwort von Prof. Dr. Erich Ortner

Deutscher Universitäts-Verlag

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.

Dissertation Technische Universität Darmstadt, 2007

D 17

1. Auflage Juli 2007

Lektorat: Frauke Schindler / Stefanie Loyal

Der Deutsche Universitäts-Verlag ist ein Unternehmen von Springer Science+Business Media.
www.duv.de

Umschlaggestaltung: Regine Zimmer, Dipl.-Designerin, Frankfurt/Main
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier
Printed in Germany

ISBN 978-3-8350-0840-3

Für meine Familie

Geleitwort

Bei der Entwicklung der Fächer „Informatik“ und „Wirtschaftsinformatik“ wird in letzter Zeit ein existenzielles Daseinsgerangel zwischen beiden immer offenbarer. Während sich die quantitativ sehr viel stärker vertretenen Informatiker zusehends dem Aufgabengebiet „Entwicklung von Anwendungssystemen“ zuwenden, bleibt für die an deutschen Universitäten zahlenmäßig unterlegenen Wirtschaftsinformatiker nur noch das „Informationsmanagement“ als ein zu besetzendes Forschungsgebiet übrig. Hinzu kommt die Tatsache, dass die Ergebnisse der Informatik-Forschung im Sinne der Computer Science in den letzten Dekaden – getrieben durch das Moore’sche Gesetz – weltweit sehr viel mehr für Fortschritt und Innovationen gesorgt haben, als dies leider auf Seiten der Wirtschaftsinformatik (Information Systems Science) der Fall war. Datenbank-Management-Systeme, objektorientierte und komponentenbasierte Entwicklung, Workflow-Management-Anwendungen, Service-orientierte Architekturen (Application Server für die Business Logic), Web 2.0-Technologien im Hinblick auf e-Commerce oder jede beliebige Lebenswelt im World Wide Web, sind nur einige Etappen, die bereits von der Anwendungs- und Systeminformatik dominiert werden.

Dass die Wirtschaftsinformatik auf dem ihr verbliebenen Terrain dennoch zu ökonomisch bedeutsamen Forschungsergebnissen kommen kann, wird in der vorliegenden Arbeit von Jörg Lonthoff demonstriert. Externes Anwendungsmanagement ist ein dem Informationsmanagement zurechenbares Forschungsgebiet, für das Herr Lonthoff organisatorische, werkzeugseitig unterstützte und geschäftspolitisch wettbewerbswirksame Konzepte sowie Implementierungen liefert. Dabei kommen in gekonnter Weise Mobilgeräte und mobile Technologien zum Einsatz.

Die Arbeit von Herrn Lonthoff bringt uns dem Ziel, Anwendungssysteme service- oder komponentenbasiert professionell managen zu können, ein großes Stück näher. Dies ist vor dem Hintergrund eines sich Bahn brechenden Service-Marktes und –Handels (Software as a Service – SaaS) auch ökonomisch von hoher Brisanz und Tragweite. Die Arbeitsergebnisse besitzen darüber hinaus kreative konstruktive Anteile, die zeigen, dass Darmstädter Wirtschaftsinformatiker auch in der Anwendungsentwicklung mit exzellenten Forschungsresultaten auf sich aufmerksam machen.

Die Lektüre dieses Buches ist somit Wirtschafts- und Anwendungsinformatikern an den Hochschulen sowie in der Wirtschaft und Verwaltung gleichermaßen zu empfehlen.

Prof. Dr. Erich Ortner

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet Wirtschaftsinformatik I – Entwicklung von Anwendungssystemen der TU Darmstadt. An dieser Stelle möchte ich mich herzlich bei den Personen bedanken, die zum Gelingen meiner Arbeit beigetragen haben.

Zuallererst möchte ich meinem Doktorvater, Herrn Prof. Dr. Erich Ortner, dafür danken, dass er mich ermutigt hat, nach meiner Diplomarbeit eine Promotion anzustreben und mir diese Entscheidung durch das Angebot eines spannenden und aktuellen Themas leicht gemacht hat. Ich möchte ihm auch dafür danken, dass ich ein hohes Maß an zeitlicher und gestalterischer Flexibilität erhalten habe. Durch ihn habe ich meine Freude an Forschung und Lehre entdeckt.

An zweiter Stelle bedanke ich mich bei Frau Prof. Dr. Susanne Strahringer für die Bereitschaft das Zweitgutachten zu übernehmen sowie für die konstruktiven Vorschläge zur Reifung dieser Arbeit.

Ganz herzlichen Dank meinen Kollegen Marcus Elzenheimer, Tobias Grollius und Joachim Sternhuber, die mit mir durch „dick“ und „dünn“ gehen, mich durch Motivation und fachliche Diskussion tatkräftig unterstützt und in der Endphase den Rücken freigehalten haben sowie die gesamte Arbeit korrekturlesen durften. Ein weiterer Dank geht an unsere gute Seele, Frau Lange, die mich während der Promotionszeit immer wieder aufgemuntert und mit Süßigkeiten versorgt hat.

Einen großen Dank richte ich an alle meine Seminar-, Studien- und Diplomarbeiter sowie die Studierenden, die im Rahmen eines Wirtschaftsinformatikpraktikums einen Prototypen für den von mir konzipierten mobilen Marktplatz entwickelt haben.

Abschließend ein persönlicher Dank an meine Familie. Meine Eltern haben mein Studium zu weiten Teilen mitfinanziert und mir das Umfeld bereitet, diese Laufbahn einschlagen zu können. Meiner Frau Silke danke ich von ganzem Herzen für ihre Geduld, mich in der Promotionszeit zu ertragen und für die Akzeptanz der „klassischen Rollenverteilung“, die wesentlich dazu beigetragen hat, dass ich mich – trotz meiner lie-

ben aber durchaus lebhaften Töchter Noreen und Pheline – voll auf die Anfertigung meiner Dissertation konzentrieren konnte.

Zu guter Letzt danke ich all denjenigen, die jetzt nicht namentlich erwähnt wurden, aber ebenfalls zum Gelingen dieser Dissertation beigetragen haben oder mich einfach nur während der Promotionszeit – manchmal bestimmt etwas gereizt – ertragen haben.

Jörg Lonthoff

Inhaltsübersicht

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ANSI	American National Standards Institute
API	Application Programming Interface
ASP	Application Service Providing
CLI	Common Language Infrastructure
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DBMS	Datenbank-Management-System
DCOM	Distributed Component Object Model
DIN	Deutsche Industrie Norm
DTD	Document Type Definition
E-NOgS³	Electronic New Organon {Server|Servant|Service}
ECMA	European Computer Manufacturers Association
EJB	Enterprise Java Beans
FTP	File Transfer Protocol
GI	Gesellschaft für Informatik e.V.
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IRDA	Infrared Data Association
ISO	International Standardization Organization
IT	Informationstechnologie
J2EE	Java 2 Enterprise Edition

J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
Jini	Java Intelligent Network Infrastructure
LAN	Local Area Network
MAN	Metropolitan Area Network
MC	Mobile Computing
MIDP	Mobile Information Device Profile
MPVM	Multipfad-Vorgehensmodell
NC	Nomadic Computing
OCL	Object Constraints Language
OMG	Object Management Group
PAN	Personal Area Network
PC	Personal Computer
PDA	Personal Digital Assistant
RMI	Remote Methode Invocation
RPC	Remote Procedure Call
SDP	Service Discovery Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPARC	Standards Planning and Requirements Committee
SQL	Structured Query Language
SSL	Secure Socket Layer
TCO	Total Cost of Ownership
TCP	Transmission Control Protocol
UC	Ubiquitous Computing
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UMTS	Universale Mobile Telecommunication System
VM	Virtual Machine

W3C	World Wide Web Consortium
WAN	Wide Area Network
WAP	Wireless Application Protocol
WfMS	Workflow-Management-System
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WTP	Wireless Transaction Protocol
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

1 Einleitung

1.1 Motivation

Die Entwicklungen auf dem im Vergleich zu anderen Wissenschaften noch jungen Gebiet der Informatik (Systeminformatik) und Wirtschaftsinformatik (Anwendungsinformatik) führen stetig zu Neuerungen (Innovationen), Weiterentwicklungen, manchmal aber auch lediglich zu neuen Schlagwörtern. Die Wirtschaftsinformatik erschließt (neue) Anwendungsgebiete, der durch die Informatik bereiteten technologischen Grundlagen. Man kann heute umfassend von einer Systeminformatik als Technologielieferant und einer Anwendungsinformatik (z. B. Wirtschaftsinformatik) als Technologieverwerter sprechen[1].

Mark Weiser prägte 1991 den Begriff „Ubiquitous Computing[2]“, womit er in akademisch-idealistischer Weise eine unaufdringliche, humanzentrierte Technikvision beschrieb[3]. Ein derartiges Ubiquitous Computing, in dem die Technik soweit in den Hintergrund tritt (Disappearing Computing[4]), dass der Mensch nur noch die Anwendung (Unterstützung) wahrnimmt, sollte Ziel der Entwicklung auf den Gebieten der Systeminformatik und Anwendungsinformatik sein. Die Idee der Ubiquität der Technologie führt zum Begriff der Omnipräsenz ihrer Anwendungssysteme.

Mit dem Konzept des Ubiquitous Computing wird ein Paradigmenwechsel eingeleitet. Der Begriff „Paradigmenwechsel“ wurde von Thomas Kuhn wie folgt definiert:

„Fortschritt in der Wissenschaft vollzieht sich nicht durch kontinuierliche Veränderung, sondern durch revolutionäre Prozesse. Ein bisher geltendes Erklärungsmodell wird verworfen und durch ein anderes ersetzt.[5]“

In den Anfangszeiten der „Elektronischen Datenverarbeitung“ (EDV) war der Mainframe (Großrechner) das Mittel zur automatisierten Massendatenverarbeitung, der von mehreren Anwendern bedient werden konnte. In den späten 1980ern wurden Rechner

---

1 Vgl. Bienert 1998, S. 14.
2 Vgl. Weiser 1991.
3 Vgl. Mattern 2003, S. 4.
4 Vgl. Roth 2005, S. 4.
5 Kuhn 1967.

für den persönlichen Gebrauch konstruiert. Diese Ära bezeichnete man als die „PC-Ära“. Mit diesen Endgeräten wurde primär das Ziel der Informationsbeschaffung sowie des Informationszugangs verfolgt und es sollte die „desktop-Verarbeitung“ (Metapher) realisiert werden. Durch Ubiquitous Computing wird potenziell jeder Mensch der Welt, also ca. 6,5 Mrd. Menschen, Informations- und Kommunikationstechnologie nutzen können, um Unterstützung zur Bewältigung des Lebens zu erhalten[6]. Aufgrund dieser grundlegenden Veränderungen (Revolutionen) des Einsatzes von Technik und der Art der Nutzung ist im Bereich des Ubiquitous Computing der Begriff Paradigmenwechsel gerechtfertigt. Die Unterstützung, die Ubiquitous Computing leisten soll, kann auf nahezu allen Gebieten erfolgen: dem Beruf, bei der häuslichen Arbeit oder generell im Alltag eines Menschen, also auch bei der Freizeitgestaltung.

So wie der Begriff „Ubiquität“ einen Paradigmenwechsel (Revolution) im Hinblick auf die Verbreitung einer Technologie für Menschen kennzeichnet, charakterisiert der Begriff „Omnipräsenz“ einen Paradigmenwechsel (Revolution) im Hinblick auf die Unterstützung des Denkens und Handelns der Menschen zur Bewältigung ihres Lebens. Ortner nennt die gesamten Einrichtungen dieser Unterstützung des Denkens und des Handelns der Menschen Anwendungssysteme (siehe Abschnitt 2.2)[7].

1967 wurde von Gordon Moore, dem damaligen Chef von Intel, das sogenannte „Moore’ sche Gesetz der Halbleitertechnologie“ aufgestellt, das eine Verdoppelung der Verarbeitungsgeschwindigkeit und der Speicherkapazität der Bauelemente der Computer-Industrie alle 18 Monate voraussagt. Mit diesem „Gesetz“ wurde exponentielles Wachstum prognostiziert. Bis heute gilt dieses „Gesetz“. Dieses technologiegetriebene Wachstum bezüglich der Leistungsfähigkeit und Speicherkapazität von Halbleiterelementen zieht ein Wachstum des Bedarfs an Software und Anwendungssystemen nach sich. Die gewonnenen technischen Möglichkeiten, z. B. unter Nutzung aller menschlicher Sinne mittels Kommunikationssystemen kommunizieren zu können, beschreiben Bell und Gray in ihrem Beitrag „The Revolution Yet to Happen“[8] als den Durchbruch der Informatik mit Blick auf die Zukunft der nächsten 50 Jahre.

Heute geht es in der System- und Anwendungsinformatik nicht mehr primär nur um Massendatenverarbeitung, sondern um die rechnerunterstützte tägliche Lebensbewälti-

---

6 Vgl. Ortner 2005a, S. 184.
7 Vgl. Ortner 2005a, S. 9f.
8 Bell; Gray 1997, S. 5.

gung der Anwender „überall“ (im Cyberspace)[9]. Die nachfolgende Abbildung zeigt zusammenfassend die Entwicklungsschritte der Informations- und Kommunikationstechnologie in Beziehung zur primären Nutzung der Technologie in der Gestalt von Anwendungssystemen. Dabei geht die Entwicklungsrichtung mit der Winkelhalbierenden einher. Der mit „HEUTE“ markierte Bereich bezeichnet den aktuellen Stand nach der Einschätzung des Autors.

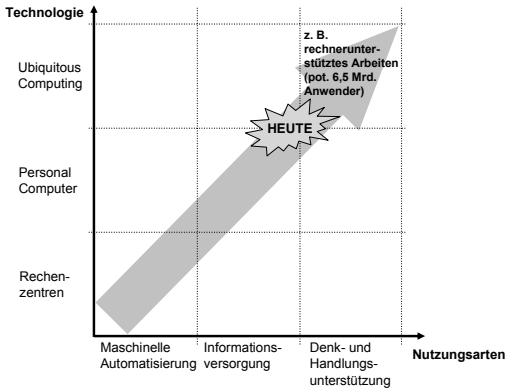

Abbildung 1: Entwicklung der Informations- und Kommunikationstechnologie[10]

Ein anhaltender Trend, der durch neue Informations- und Kommunikationstechnologie revolutioniert wird, ist die Mobilität des Menschen. Mobilität heißt heute nicht mehr nur, von einem Ort A zu einem Ort B zu pendeln. Vielmehr bedeutet Mobilität, dass man zeit- und ortsunabhängig seinen Tätigkeiten oder aber auch privaten Verpflichtungen nachgehen kann. Mobilität schließt auch geistige Mobilität[11] und damit Wissensmobilität ein. Auch die geistige Mobilität, bspw. des sogenannten Wissensarbeiters, erfährt in unserer Zeit einen revolutionären Wandel, denn auf dem Weg zur globalen Wissensgesellschaft gehört das sogenannte „lebenslange Lernen[12]“ und der globale Wettbewerb mit anderen Wissensarbeitern zum Alltag. Nur wer zu jeder Zeit an jedem Ort in der Lage ist, relevantes Wissen, das in Form von Schemata akkumuliert wird, aufzunehmen bzw. zu aktualisieren, kann sich in der „Wissensgesellschaft“ bewähren.

---

9 Vgl. Grollius; Lonthoff; Ortner 2006a, S. 363.
10 In Anlehnung an Brandt; Lonthoff 2004, S.34.
11 Vgl. Lonthoff 2004, S. 452.
12 Vgl. Kruse 2003.

Auch hier kommt Weisers Vision des Ubiquitous Computing, in dem die Menschen orts- und zeitunabhängig durch Informations- und Kommunikationstechnik, z. B. aus den Bereichen des „Mobile Computing“ (Enterprise Mobility/Human Mobility[13]), omnipräsent unterstützt werden, zum Tragen. Sogar „Denkunterstützung“ in umfassendster Form ist durch Methoden des Wissensmanagements (basierend auf Sprachtheorie und Logik) sowie den Einsatz von mobilen Endgeräten für den Menschen heute überall vorstellbar[14].

1.2 Problemstellung

Wenn nun der Mensch umfassend durch Technologie unterstützt werden soll und gleichzeitig alle Menschen der Welt als potenzielle Anwender (Nutzer) angesprochen werden, so bedingt dies, dass die Art der Unterstützung intuitiv erlernbar und einfach nutzbar ist. Auch hier ist eine Entwicklung notwendig, weg vom „Bedienen“ eines Gerätes, hin zur Interaktion mit einer Anwendung.

Hierbei hat sich ein großes Forschungsfeld für die Wissenschaft sämtlicher Disziplinen ergeben. Denn mit der Forderung nach intuitiver, einfacher Interaktion mit Anwendungen für jeden Menschen erschließen sich Problemfelder auf der Seite der Technik (Informatik, Elektrotechnik, Mechanik etc.) bis hin zu der Seite der Menschen (Psychologie, Ergonomie, Soziologie, Medizin etc.). Im Bereich der Technik müssen technische Heterogenitäten überwunden werden, die aus einer hohen Vielfalt unterschiedlichster Hardware und Software (Oberflächen, Protokolle etc.) resultieren. Gerade im Umfeld spontaner Vernetzung von Ressourcen (ad hoc-Netzwerke) ergeben sich viele Probleme. Im Umfeld des Anwenders sind Aspekte menschlicher Wahrnehmung von Interesse, wenn es um Mensch-Technik-Interaktion geht. Auch gesellschaftliche Auswirkungen des Technologieeinsatzes sollten abgeschätzt werden.

An der Schnittstelle zwischen Mensch und Technik forscht eine interdisziplinär aufgestellte Wissenschaftsdisziplin „Human Computer Interaction“[15] (HCI) an besseren Zugängen der beteiligten Akteure „Mensch“ und „Maschine“. Hierbei ist gerade im mobilen Bereich[16] der Trend erkennbar, weg von der Handeingabe und Bildausgabe hin zu Spracheingabe und -ausgabe. Sogar der haptische und olfaktorische Bereich sowie

[13] Vgl. Lonthoff 2007.
[14] Vgl. Heinemann 2006, S. 135.
[15] Vgl. Dix 2004.
[16] Vgl. Chittaro 2003.

die Muskelkontraktion verschiedener Körperstellen des Menschen werden als Interaktionsmedium erforscht. Usability Engineering[17] fasst solche Bemühungen zusammen, deren Ziel die Bereitstellung von Werkzeugen und Methoden für das Entwickeln gebrauchstauglicher Anwendungssysteme ist, mit denen die Menschen interagieren, statt sie nur zu bedienen.

Der Autor möchte mit den Methoden und Werkzeugen der Anwendungs-, speziell der Wirtschaftsinformatik, eine Systematik (und Werkzeuge) des externen Anwendungsmanagements vorstellen, die den Anwender von Tätigkeiten der Planung, Steuerung und Kontrolle der Anwendungen, die er nutzen möchte, entlastet.

Ein weiters Problemfeld ergibt sich bei der Auswahl der Zielgruppe, die durch (mobile) Rechner in ihrem Handeln unterstützt werden soll: Wissensarbeiter (durch Informatisierung der Arbeit[18]) und darüber hinaus potenziell alle Menschen (durch Unterstützung des Lebens). Bei dieser einschränkungsfreien Zielgruppe der Anwender darf spezielles IT-Wissen nicht vorausgesetzt werden. Wohl aber wäre zu wünschen, dass dieser Personenkreis über eine hohe Sprachkompetenz verfügt. Aber selbst auf diesem Feld ist heute Rechnerunterstützung und Abhilfe möglich[19].

1.3 Zielsetzung

Das Ziel dieser Dissertation besteht in einem konstruktiven konzeptionellen Aufbau eines externen Anwendungsmanagement für omnipräsente Anwendungssysteme und einer prototypischen Implementierung einiger Management-Instrumente. Dabei soll die Entwicklung eines solchen Managementsystems menschorientiert (anthropozentrisch) erfolgen, da bei dieser Aufgabe immer noch Tätigkeiten vom Menschen ausgeführt werden müssen sowie der Mensch in erster Linie bei seinem Denken und Handeln durch die Technologie unterstützt werden soll.

Basierend auf dem Komponenten-Paradigma: *„Die Teile werden aus einem Katalog entnommen und dann nach einem Plan zusammengesetzt.*[20]*“*, wird von der komponentenbasierten Anwendungssystementwicklung gebrauch gemacht. Das Anwendungsmanagement soll hierbei den gesamten (Software-)Komponenten-Lebenszyklus unter-

[17] Vgl. Nielsen 2003.
[18] Vgl. Rürup 1998.
[19] Vgl. Heinemann 2006, S. 132f.
[20] Ortner 2005a, S. 115.

stützen. In der vorliegenden Arbeit wird der Fokus auf den für das externe Anwendungsmanagement relevanten Teil des Lebenszyklus eines komponentenbasierten Anwendungssystems, also die Phase der externen Bereitstellung von (Software-)Komponenten durch Dritte, gelegt. Die Phasen „Einsatz“ und „Dekonstruktion“ von (Software-)Komponenten bei flexiblen, ad-hoc zusammengestellten Anwendungssystemen werden zum sogenannten „internen Anwendungsmanagement“[21] gerechnet.

Ausgehend von dieser Zielsetzung werden zwei zentrale Forschungsfragen formuliert, die wiederum in Unterziele gegliedert sind. Dabei werden organisatorische (Management), fachliche (Anwendungsgebiete) und technische (Hardware/Software) Aspekte betrachtet.

1. Wie ist ein lebenszyklusorientiertes externes Anwendungsmanagement für mobile verteilte Systeme zu gestalten?
    a. Welche Aufgaben des Anwendungsmanagements lassen sich für mobile verteilte Systeme identifizieren (organisatorisch)?
    b. Wie lässt sich ein Anwendungsmanagement lebenszyklusorientiert unterteilen (organisatorisch)?

2. Wie gelingt ein Handel heterogener Komponenten für (mobile) Anwender ohne Einführung einer spezifischen Middleware?
    a. Wodurch wird ein elektronischer Marktplatz mobil verfügbar (technisch)?
    b. Wodurch wird eine Software-Komponente für mobile Endgeräte elektronisch vermarktbar (organisatorisch/fachlich/technisch)?
    c. Ist eine einheitliche Komponentenspezifikation in einem Prototypen implementierbar (technisch)?
    d. Welchen Beitrag leistet ein elektronischer Komponenten-Marktplatz im Rahmen eines externen Anwendungsmanagements (technisch, fachlich, organisatorisch)?

Ein weiterer Lösungsansatz und Anspruch dieser Arbeit ist die Beschreibung des mobilen (Software-)Komponenten-Marktplatzes mobiCOMP als ein Prototyp und Teil eines Systems für das externe Anwendungsmanagement von (Software-)Komponenten für komponentenbasierte Anwendungssysteme im mobilen Umfeld.

---

[21] Vgl. Grollius; Lonthoff; Ortner 2006b, S. 121.

Im Gegensatz zu anderen Ansätzen, die eine neue Infrastruktur für omnipräsente Anwendungssysteme definieren (wie z. B. in Modahl et al.[22] aufgezählt), soll das im Rahmen dieser Arbeit zu entwickelnde Konzept die Ankopplung existierender Technologien und damit eine gesamtheitliche Integration der vorhandenen Ressourcen ermöglichen. Durch den ganzheitlichen Ansatz von Anwendungssystemen (siehe Abschnitt 2.2) werden noch andere Ressourcenbereiche, wie z. B. Prozesse, Wissen und Menschen als „Teile“ von Anwendungssystemen dynamisiert. Dies ermöglicht neue Formen der Arbeitsorganisation in Netzen, wie sie durch Service-orientierte Architekturen (SOA) und Web 2.0 – ein Marketingbegriff für das Semantic Web – heute diskutierbar sind.

1.4 Wissenschaftstheoretischer Standpunkt

Da diese Arbeit die ingenieurmäßige Entwicklung eines Anwendungssystems zum Gegenstand hat, werden das Vorgehen und die Methoden des Erlanger Konstruktivismus (im Folgenden nur kurz „Konstruktivismus“ genannt) eingesetzt. Der Erlanger Konstruktivismus (auch Konstruktivismus der Erlanger Schule, später Methodischer Konstruktivismus) ist ein Ansatz einer allgemeinen Wissenschaftstheorie[23].

Der Konstruktivismus hat es sich zur Aufgabe gemacht, die Erzeugung der Gegenstände einer Wissenschaft durch die Befolgung ausdrücklicher und klar nachvollziehbarer Vorschriften zu (re-)konstruieren. Zu beachten ist, dass hierbei eine Axiomen- und Prototypenfreiheit eingehalten wird. Alle Elemente und Regeln der Wissenschaftssprache sollen voraussetzungsfrei, zirkelfrei und nachvollziehbar eingeführt werden[24]. Heinemann leitet daraus wie folgt das Grundpostulat konstruktivistischer Rekonstruktionsprozesse ab: stets *„alles explizit machend, schrittweise und zirkelfrei*[25]*“* entwickeln.

Ein Grundlagenwerk des Konstruktivismus ist die „Logische Propädeutik[26]“ von Kamlah und Lorenzen. In ihrem Buch beschreiben beide Autoren, wie man sprachlichen Missverständnissen und Unschärfen mittels einer klaren, zirkelfreien, Wissenschaftssprache, die bspw. auch als Konstruktionssprache dienen kann, vorbeugen sollte:

---

[22] Modahl et al. 2006.
[23] Vgl. Frank 2006, S. 19.
[24] Vgl. Mittelstraß 2004, S. 551.
[25] Heinemann 2006, S. 9 und 40.
[26] Kamlah; Lorenzen 1996.

„An den Terminus als Prädikator einer wissenschaftlichen Sprache stellen wir folgende Anforderungen: Die Verständigung zwischen den Gesprächspartnern soll nicht dadurch beeinträchtigt werden, dass der Redende den Prädikator anders verwendet als der Hörende (umgangssprachlich ausgedrückt: dass sich der Hörende ‚etwas anderes dabei denkt' als der Redende). Um dieses Ziel zu erreichen, werden die Gesprächspartner vor der Verwendung eines Terminus gut daran tun, sich hinsichtlich eben dieser Verwendung ausdrücklich zu verständigen.[27]*"*

Solche Prädikatoren-Systeme oder Terminologien werden in einem besonderen Zweig der Informatik, die man „Künstliche Intelligenz" nennt, heute Ontologien genannt.

Der Konstruktivismus steht seit der sprachkritischen Wende (linguistic turn[28]) etwa Mitte des 20. Jahrhunderts in engem Zusammenhang mit der modernen Sprachphilosophie. Wissenschaft wird als zweckgerichtetes Handeln verstanden. Dadurch wird ein pragmatischer Ansatz geschaffen, der in seiner Handlungstheorie zentral Sprachhandlungen einbezieht. Der Konstruktivismus orientiert sich also nicht an stillschweigenden Prämissen und Axiomen, sondern ist an den Kontext bzw. Ko-Text[29] und den Alltag (Leben) der an der Handlung Beteiligten gebunden. Im Rahmen der konstruktivistischen Methodologie werden Begriffe dialogisch eingeführt, also (re-)konstruiert, geprüft und schließlich als eindeutig nachvollziehbare Fachbegriffe etabliert, die dann in einer normierten Fachsprache – Lorenzen nennt sie Orthosprache – nach den Regeln der Aussagen- und Prädikatenlogik i. w. S. jedoch genauer nach den Regeln einer rationalen Syntax und Semantik i. e. S. verwendet werden können[30].

Traditionell beschäftigt man sich in der Anwendungs- und Systeminformatik mit dem Gegenstand „Information". Da alle Artefakte während der Entwicklung von Anwendungssystemen sprachliche Konstrukte sind, ergibt sich eine Erweiterung der Sichtweise im sprachbasierten Ansatz der Anwendungs- und Systeminformatik durch den Gegenstand „Sprache"[31]. Mit diesem Ansatz kann die Sprachtheorie (Sprachwissenschaft), neben den Disziplinen der Ingenieurwissenschaften, der Mathematik und Lo-

---

[27] Kamlah; Lorenzen 1996.
[28] Vgl. Rorty 1967.
[29] Nach Bar-Hillel bezeichnet der Begriff „Ko-Text" speziell die sprachliche Umgebung eines sprachlichen Ausdrucks in Abgrenzung zum Begriff „Kontext", der sowohl für die sprachliche als auch außersprachliche Umgebung verwendet wird (vgl. Mittelstraß 2004, Stichwort „Kotext").
[30] Vgl. Schienmann 1997, S. 15.
[31] Vgl. Ortner 1993, S. 7f.

gik, zur erforderlichen theoretischen Fundierung der Anwendungs- und Systeminformatik verwendet werden.

Ausgehend von dieser Sprachbasierung, die auch eine Weiterentwicklung der Sprechakttheorie Austins[32] darstellt, kann die Anwendungs- und Systeminformatik als neues Ingenieurfach zur Entwicklung von Sprachartefakten für alle Subjekte, Gegenstände oder Systeme, die über Sprache überhaupt erreichbar sind, bezeichnet werden[33].

1.5 Aufbau der Arbeit

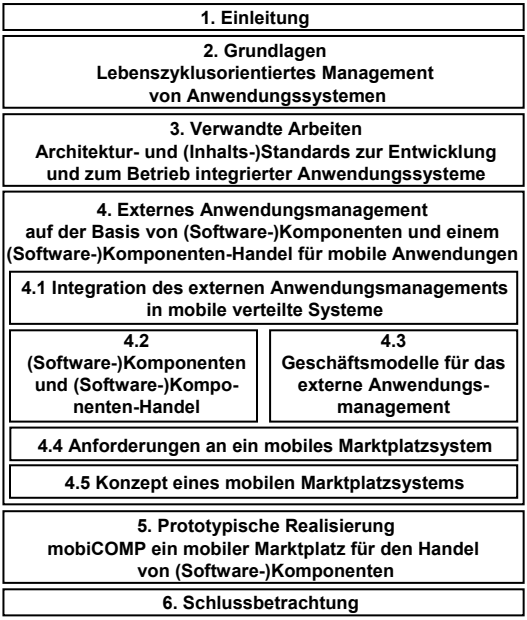

Abbildung 2: Kapitelaufbau der Arbeit

Nach diesen einführenden Bemerkungen zur Thematik des externen Anwendungsmanagements werden im Kapitel 2 die Grundlagen für das lebenszyklusorientierte Management von Anwendungssystemen behandelt. Kapitel 3 stellt im Umfeld der Architektur- und (Inhalts-)Standards zur Entwicklung und zum Betrieb integrierter Anwendungssysteme verwandte Ansätze vor. Dieses Kapitel gibt einen breiten Überblick über relevante Arbeiten zur Realisierung eines externen Anwendungsmanagements.

[32] Austin 1962.
[33] Vgl. Ortner 2005a, S. 5.

Aufbauend auf diesen Grundlagen und Ansätzen wird in Kapitel 4 ein Konzept für das externe Anwendungsmanagement auf der Basis von (Software-)Komponenten und einem (Software-)Komponenten-Handel für mobile Anwendungen aufgebaut. Als Teilergebnis eines externen Anwendungsmanagements wird ein Referenzmodell für ein mobiles Marktplatzsystem zum Handel von (Software-)Komponenten konstruiert. In Kapitel 5 wird dann mit mobiCOMP eine implementierte, prototypische Teilrealisierung eines mobilen Marktplatzsystems für den Handel von (Software-)Komponenten vorgestellt. Abschließend werden im Kapitel 6 Schlußbetrachtungen der Arbeit getroffen. Abbildung 2 fasst den Kapitelaufbau der Arbeit grafisch zusammen.

2 Lebenszyklusorientiertes Management von Anwendungssystemen

Durch das Anwendungsmanagement sollen Probleme der Planung, Steuerung und Kontrolle von Anwendungen in ihrem Gebrauch gelöst werden[34]. Hierzu werden zunächst die Begriffe „Anwendung“ und „Anwendungssystem“ definiert. Im Folgenden wird der Lebenszyklus[35] von Anwendungssystemen aus zwei Perspektiven aufgezeigt: aus Sicht des Anwendungssystems und aus Sicht seiner Bestandteile (Komponenten). Aufbauend auf Managementaspekten aus der Anwendungsinformatik wird eine Definition des Anwendungsmanagements hergeleitet. Visionär betrachtet können Aufgaben des Anwendungsmanagements durch ein Anwendungssystem-Management-System, das in Grollius et al.[36] vorgestellt wird, rechnerunterstützt automatisiert werden. Darauf aufbauend werden Aspekte des Anwendungsmanagements für mobile Anwendungssysteme beschrieben. In Anlehnung an die Unterscheidung von Betriebs- und Geschäftsprozessen und unter Einordnung des Anwendungsmanagements in den Lebenszyklus von Anwendungssystemen, wird das Anwendungsmanagement in ein internes und ein externes Anwendungsmanagement unterschieden. Diese Arbeit betrachtet insbesondere das externe Anwendungsmanagement.

2.1 Anwendung

Der Begriff der „Anwendung“ (engl. „Application“) kann nach dem ANSI Telecom Glossary 2000 definiert werden als: *„Software that performs a specific task or function, such as word-processing, creation of spreadsheets, generation of graphics, facilitating electronic mail etc. Synonym application software.*[37]*“*

Im Unterschied zu dieser Definition unterscheidet Ortner feiner zwischen spezifischer Anwendung (Application) oder synonym Anwendungssoftware (Application Software) auf der einen Seite und generischen Anwendungen oder synonym System- bzw. Basissoftware auf der anderen Seite[38]. Mit „Anwendung“ ist folglich Software gemeint, die eine bestimmte fachliche und spezifische Funktionalität zur Erfüllung einer Aufgabe bereitstellt und damit auf die Verarbeitung von Informationen besonderen

---

[34] Vgl. Ortner 1991a, S. 321.
[35] Die Begriffe „Anwendungsmanagement“ und „Lebenszyklus“ werden hierbei im Sinne der Informatik verwendet. In der Wirtschaftsinformatik werden diese Begriffe darüber hinaus im Sinne einer Produktlebenszyklus-Planung gebraucht, wie bspw. in Lehner 1989 beschrieben.
[36] Vgl. Grollius et al. 2005.
[37] ANSI 2001.
[38] Vgl. Ortner 1991a, S. 321.

Inhalts zweckgerichtet ist. Eine solche Anwendung positioniert sich in der Informationsverarbeitung eines Unternehmens „über" den Daten und der Basissoftware und bettet sich „nach oben" in die Prozesse einer Organisation ein (siehe Abbildung 3).

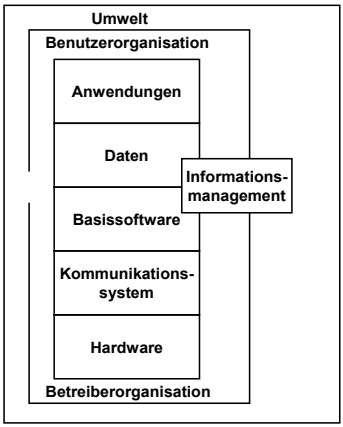

Abbildung 3: Schichtung der Anwendungssysteme im Informationsmanagement[39]

Beispiele einer Anwendung können sowohl eine Software zur Tabellenkalkulation wie auch eine Kundenadressverwaltung sein. Eine Anwendung kann aber auch über die Grenzen eines Computers hinausgehen. So ist eine unternehmensweite Auftragsverwaltung, die verteilt auf mehreren Rechnern betrieben wird, ebenfalls eine Anwendung.

Während mit dem Begriff „Anwendung i. e. S." die funktionale Steuerungs- bzw. Ablauflogik (auch Geschäftslogik genannt) bezeichnet wird, umfasst z. B. bei Datenbankanwendungen der Begriff „Anwendungssoftware" die beiden Ebenen „Anwendung" und „Daten". Mit einer Anwendung kann dann auf Daten operiert werden. Eine Buchhaltungssoftware mit den Buchungsdaten ist somit ein Beispiel für die Anwendungssoftware eines Unternehmens.

2.2 Anwendungssystem

Unter einem Anwendungssystem[40] wird ein mit rechnerunterstützter Symbolverarbeitung versehener Gegenstand jeglicher Art, der sich im Gebrauch befindet, zusammen

[39] In Anlehnung an Ortner 1991a, S. 321.
[40] Vgl. Wedekind 1973.

mit seiner Umgebung verstanden[41]. Dabei betont der Begriff „Anwendungssystem“ den ganzheitlichen Ansatz. Die Abbildung 4 zeigt ein komponentenbasiertes Ebenenmodell der Anwendungssysteme nach Ortner. In diesem Ebenenmodell umfasst der Begriff „Anwendungssystem“ alle aufgezeigten Ebenen.

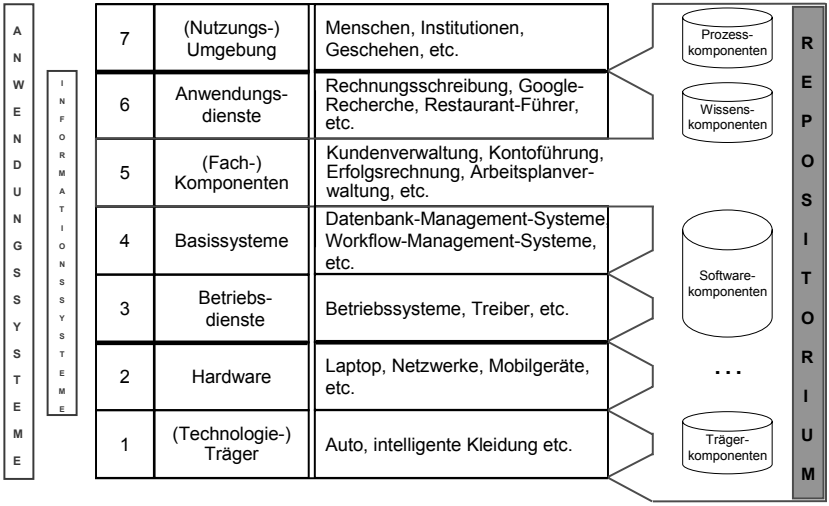

Abbildung 4: Komponentenbasiertes Ebenenmodell der Anwendungssysteme[42]

Lässt man bei der Einordnung der Begriffe die Organisation (Ebene 7) sowie den Technologieträger (Ebene 1) weg, so spricht man von einem Informationssystem. In großen Teilen der Wirtschaftsinformatik-Literatur[43,44,45,46,47,48] werden die Begriffe „Informationssystem“ und „Anwendungssystem“ sehr unterschiedlich[49], manchmal auch gerade umgekehrt verwendet, also Informationssystem als der umfassendere Begriff. Informationssysteme sind unter dem Aspekt der Anwendungssoftware ein (Forschungs-)Gegenstand der Wirtschaftsinformatik. Ein anderer wichtiger (Forschungs-)Gegenstand der Wirtschaftsinformatik ist das Informationsmanagement. Ort-

41 Vgl. Ortner 2005a, S. 34.
42 In Anlehnung an Ortner 2003.
43 Vgl. Krcmar 2000, S.20.
44 Vgl. Fink; Schneidereit; Voß 2001, S. 3 und 189.
45 Vgl. Ferstl; Sinz 2001, S. 4.
46 Vgl. Stahlknecht; Hasenkamp 2005, S. 204.
47 Vgl. Hansen; Neumann 2005, S. 84.
48 Vgl. Mertens 2005a, S. 1.
49 Vgl. Schiemann 1992, S. 253-254.

ner bezieht den Menschen in den Entwicklungsprozess von Anwendungen ebenso mit ein, wie Technologieträger, Organisationsprozesse oder menschliches Wissen.

Hier wird deutlich, dass sich ein Anwendungssystem über drei verschiedenen Hauptebenen erstreckt: die physischen Objekte (Ebene 1 – Technologieträger und Ebene 2 – Hardware), die sprachlichen Objekte (Ebene 3 – Betriebsdienste, Ebene 4 – Basissysteme, Ebene 5 – (Fach-)Komponenten und Ebene 6 – Anwendungsdienste) sowie die Menschen, die Organisation und die Geschehen (Ebene 7 – (Nutzungs-)Umgebung). Damit wird ein Anwendungssystem aus einer Gesamtheit von physischen Dingen, immateriellen Voraussetzungen, wie Wissen (z. B. Organisationsstrukturen) und Software sowie den Anwendern (Menschen) gebildet.

Technologieträger können „computerisierte“ (informatisierte) Gegenstände sein, d. h. Gegenstände, die mit Informations- und Kommunikationstechnik ausgerüstet sind. Auf der Ebene der Hardware sind Rechner und Netzwerkhardware einzuordnen. Betriebssoftware stellt Betriebsdienste der Hardware an die darüberliegende Ebene bereit. Generische Software stellt in Form von Basissystemen generische (wieder verwendbare) Funktionen für Anwendungen bereit. Auf der Ebene der (Fach-)Komponenten werden Software- und Wissenskomponenten von Anwendungen bereitgestellt. Die Anwendungsdienste stellen dem Anwender die geeigneten Unterstützungen bereit, in dem sie die Ablaufbeschreibungen und Kontrollstrukturen der (Fach-)Komponenten bereitstellen. Die (Nutzungs-)Umgebung ist das Umsystem in das eine Anwendung eingebettet ist, also bspw. ihre Organisation bzw. die Prozesse.

Neben dieser Ebenenaufteilung findet sich in Abbildung 4 ein Repositorium, das für die einzelnen Ebenen spezifische Komponenten bereitstellt. So können die Technologieträger selbst wiederum aus Technologieträger-Komponenten bestehen. (Fach-)Komponenten, Basisysteme und Betriebsdienste basieren auf (Software-)Komponenten. Anwendungsdienste unter Einbezug der (Nutzungs-)Umgebung basieren auf Wissenskomponenten und kommen in Prozessen, die aus Prozesskomponenten zusammengesetzt sind, zum Einsatz.

Im Rahmen der komponentenorientierten Anwendungsentwicklung findet die Entwicklungsarbeit nur auf den Ebenen 5 bis 7 statt, da die Basissysteme nicht verändert werden sollen, sondern bereits eine Plattform für komponentenorientierte Systeme be-

reitstellen. Dies sind auch zentrale Gedanken von CORBA[50] (siehe Abschnitt 3.4.4) und Service-orientierten Architekturen[51] (SOA) (siehe Abschnitt 3.6.2). Der ganzheitliche Anwendungssystemansatz umfasst hingegen die Ebenen 1 bis 7.

Als Beispiel für eine Anwendung auf einem mobilen Endgerät sei eine Auftragsverwaltung genannt. Wird die Umgebung (technisch und organisatorisch) einer Anwendung mit einbezogen, so ist die Rede von einem Anwendungssystem. Ein Navigationssystem im Fahrzeug bildet während der Fahrt ein Anwendungssystem mit dem Fahrer, dem Fahrzeug und evtl. mit einbezogenen Telematik-Systemen als Teile der Verkehrsinfrastruktur.

Weitere Beispiele sind:

- Eine Person mit einem informationstechnischen mobilen Endgerät beim Einkaufen.
- Ein Unternehmen mit seiner rechnerunterstützten Informationsverarbeitung am Markt.
- Eine Auktion bei eBay.
- Ein an das Internet angeschlossenes, bewohntes Gebäude.
- Eine mit einem Computer-Chip (unsichtbar) bestückte Milchtüte, die den Verbraucher rechtzeitig warnen kann, dass ihr Inhalt sauer zu werden droht.

Allen diesen Beispielen ist gemeinsam, dass sie Anwendungen in den Kontext der Nutzung stellen und zusammen genommen Anwendungssysteme genannt werden.

2.3 Lebenszyklus von Anwendungssystemen

Anwendungssysteme weisen typische Lebenszyklen[52] auf, die aus den Phasen Planung/Entwicklung, Herstellung, Vermarktung/Gebrauch und Abbau/Entsorgung bestehen[53].

Im Falle der komponentenbasierten Anwendungssysteme können hierbei zwei Lebenszyklen (siehe Abbildung 5) unterschieden werden[54]. Auf der einen Seite findet die Entwicklung von Anwendungssystemen (Vorgehensmodelle) und auf der anderen Sei-

[50] Vgl. OMG 2006.
[51] Vgl. Dostal et al. 2005.
[52] Vgl. Deemer 1980.
[53] Vgl. Heinrich 2002, S. 235f.
[54] Vgl. Mili et al. 2002, S. 437.

te die Konstruktion und die Dekonstruktion (Abbau) von Komponenten eines Anwendungssystems statt.

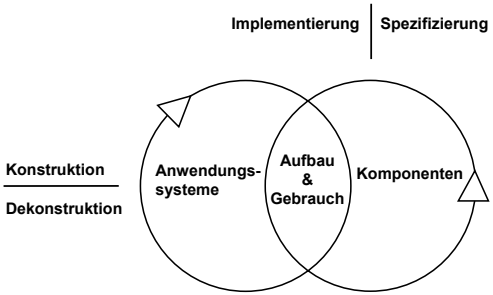

Abbildung 5: Lebenszyklus von Anwendungssystemen und Komponenten[55]

Komponenten können zunächst unabhängig von ihrer Verwendung von einem Expertentyp (z. B. UML-Modellierer) spezifiziert und von einem anderen Expertentyp (z. B. C#-Implementierer) implementiert werden. Auf der Seite der Anwendungssysteme werden die von Expertenwissen abhängigen Aufgabenkomplexe Konstruktion der Anwendungssysteme, Aufbau der Anwendungssysteme aus Komponenten, Management des Anwendungssystem-Gebrauchs sowie rückstandsfreie Dekonstruktion der Anwendungssysteme unterschieden.

2.3.1 Lebenszyklusorientiertes Vorgehensmodell der Anwendungssystementwicklung

Zur Entwicklung von Anwendungssystemen gibt es verschiedene Vorgehensmodelle. Sie unterscheiden sich meist in phasenorientierte (z. B. Wasserfallmodell[56]) und ebenenorientierte Ansätze. Daneben gibt es weitere Modelle, wie bspw. Spiralmodelle[57], aspektorientierte Modelle und andere Metaphern (Fontäne[58], Baseball[59], Fraktal[60], Whirlpool[61] etc.). Bei Phasenmodellen wird hauptsächlich der Aspekt der zeitlichen Abhängigkeit zwischen einzelnen Entwicklungsschritten betont[62]. Ebenenmodelle, auch abstraktionsebenenorientierte Modelle genannt, dienen dem Zweck der Komple-

[55] In Anlehnung an Ortner 1991a, S. 319.
[56] Vgl. Royce 1970.
[57] Vgl. Boehm 1988.
[58] Vgl. Henderson-Sellers; Edwards 1994.
[59] Vgl. Coad; Nicola 1993, S. 12.
[60] Vgl. McGregor; Sykes 1992, S. 41.
[61] Vgl. Williams 1996, S. 41.
[62] Vgl. Schienmann 1997, S. 20.

xitätsreduktion im Hinblick auf die Gesamtaufgabe und damit einer einfacheren (übersichtlicheren) Lösungsfindung[63]. Ein weiterer Aspekt der Ebenenbildung ist die Integration der Teillösungen auf verschiedenen Abstraktionsebenen, bspw. fachlich, logisch, physisch. Das Multipfad-Vorgehensmodell (MPVM) von Ortner[64] kombiniert diese beiden Sichten (siehe Abbildung 6). Es ist sowohl phasenorientiert als auch abstraktionsebenenorientiert und erlaubt darüber hinaus die Entwicklung von Anwendungssystemen unterschiedlichen Typs (z. B. Datenbank-Anwendungen, Workflow-Management-Anwendungen) mit ein und demselben Vorgehensmodell.

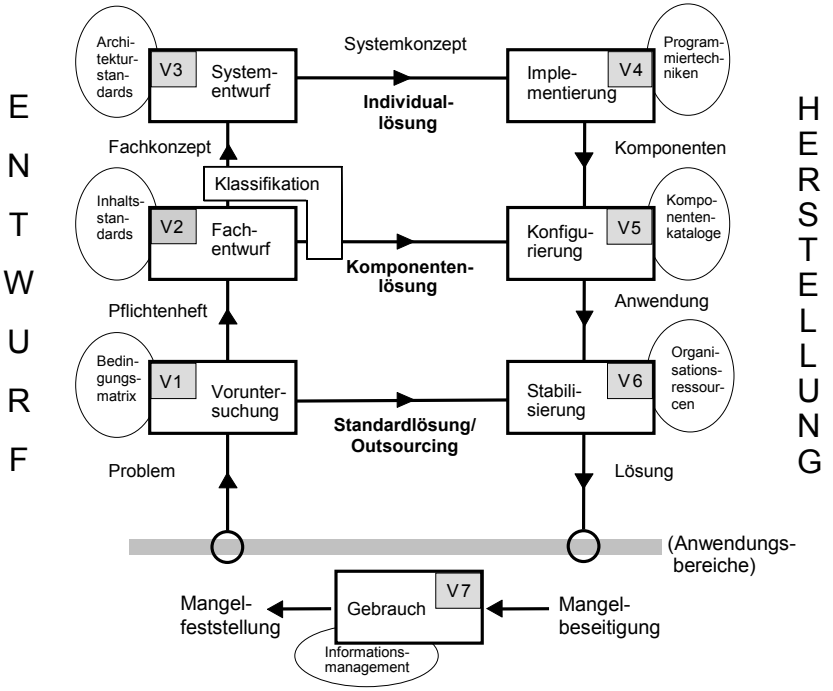

Abbildung 6: Multipfad-Vorgehensmodell für die Entwicklung von Anwendungssystemen[65]

Im MPVM (siehe Abbildung 6) werden sieben Vorgangstypen oder auch Phasen, symbolisiert durch ein großes „V“ (V1-V7), betrachtet: Voruntersuchung, Fachentwurf, Systementwurf, Implementierung, Konfigurierung, Stabilisierung und Gebrauch. Die einzelnen Phasen des MPVM lassen sich dabei grob in zwei Bereiche aufteilen, eine Entwurfsseite (linke Seite) und eine Herstellungsseite (rechte Seite).

---

[63] Vgl. Ortner 2005a, S. 46.
[64] Vgl. Ortner 2005a, S. 44ff.
[65] In Anlehnung an Ortner 2005a, S. 47.

V1 – Voruntersuchung

Nach einer Mangelfeststellung beginnt die Phase der Voruntersuchung. In dieser Phase werden die Rahmenbedingungen, das Problemfeld und mögliche Lösungswege zusammengetragen und bewertet.

Ein Hilfsmittel zu einer strukturierten Vorgehensweise bei der Voruntersuchung ist die Bedingungsmatrix[66]. In den Zeilen werden Kategorien aufgetragen, wie bspw. technisch, wirtschaftlich, normativ und menschorientiert. Die Spalten sind entsprechend der Lebensphasen für das zu entwickelnde Produkt bezeichnet, also bspw. Entwicklung, Vertrieb, Betrieb und Dekonstruktion (Abbau). Die Bedingungsmatrix als Hilfsmittel zur Formulierung aller Anforderungen liefert letztendlich die inhaltlichen Grundlagen für das Pflichtenheft.

Das Pflichtenheft ist das Ergebnisdokument der Phase Voruntersuchung und kann als ein Vertrag zwischen einem Auftraggeber und einem Auftragnehmer aufgefasst werden.

V2 – Fachentwurf

Auf den Ergebnissen der Voruntersuchung basierend, wird in der Phase Fachentwurf das fachliche Lösungskonzept entwickelt. Dabei soll das Lösungskonzept ausschließlich problemorientiert und technologieunabhängig, d. h. auch methodenneutral, formuliert sein[67]. Der Fachentwurf wird sprachlich auf Basis syntaktischer Regeln (Rationale Grammatik[68]) konstruiert, die durch die rekonstruierte Terminologie aus den Anwendungsbereichen zu einer Fachnormsprache für Anwendungssysteme erweitert wird. Damit werden im Fachentwurf sowohl die fachliche Lösung zur spezifischen Aufgabenstellung (Aussagensammlung) als auch die rekonstruierte Fachsprache des betreffenden Anwendungsbereichs als „Inhaltsstandard“ und fachliche Integrationsbasis (Unternehmensfachsprache) iterativ ermittelt. Diese Fachnormsprache muss während der Rekonstruktion von Begriffsdefekten (Synonyme, Homonyme, Äquipollenzen, Vagheiten, falsche Bezeichner) befreit und unter Einbeziehung der späteren Anwender normiert bzw. für den Gebrauch stabilisiert werden[69].

---

[66] Vgl. Wedekind; Ortner 1980, S. 27f.
[67] Vgl. Schienmann 1997, S. 24.
[68] Vgl. Lorenzen 1985, S. 13-34.
[69] Vgl. Ortner 2005a, S. 69.

Das Fachkonzept ist das Ergebnisdokument der Phase Fachentwurf. Es beschreibt das zu entwickelnde Anwendungssystem vollständig aus fachlicher Sicht in Form einer Aussagensammlung, deren Aussagen in der rekonstruierten und normierten Gebrauchs- bzw. Fachsprache der Anwender abgefasst sind. Die rekonstruierte Fachsprache garantiert die inhaltliche Integration (Inhaltsstandards) der fachlichen Lösungen.

V3 – Systementwurf

In der Phase Systementwurf wird auf der Grundlage des Fachkonzepts, abhängig von der eingesetzten Technologie, d. h. dem vorliegenden Programmierparadigma, der Programmiertechnik, der Basissoftware und den daraus resultierenden Beschränkungen, die Gesamtarchitektur (Aufbau) der Anwendung mit ihren einzelnen Komponenten logisch entworfen. Bei dem Übergang vom Fachentwurf zum Systementwurf wird über den Anwendungssystemtyp entschieden. Abhängig von diesem Anwendungssystemtyp kommen bestimmte Methoden (Vorgehensweise und (Diagramm-)Sprache) zur Modellierung zum Einsatz. Beispiele für Anwendungssystemtypen sind: Datenbank-Anwendungen, Workflow-Management-Anwendungen oder objektorientierter bzw. komponentenbasierter Entwurf von Anwendungssystemen. Je nach Anwendungssystemtyp werden die Aussagen des Fachkonzepts nach einem Klassifikationsschema (Gegenstandseinteilung) festgelegten Ergebnisbereichen zugeordnet.

Beispiele für im Software Engineering eingesetzte Diagrammsprachen sind die Unified Modeling Language[70] (UML) oder die Objekttypenmethode[71] (OTM), die Anfang der 1980er für den logischen Entwurf von Datenbank-Anwendungen[72] entwickelt wurde.

Als Integrationsmittel im Systementwurf bieten sich Formalstandards, wie bspw. die Unified Modeling Language (UML), Technologiestandards, wie bspw. eine implementierte Extensible Markup Language (XML) und Architekturstandards, wie bspw. Common Object Request Broker Architecture (CORBA) und Electronic New Organon {Server|Servant|Service}[73] (E-NOgS³) an[74].

---

[70] Vgl. Oestereich 2005.
[71] Vgl. Ortner; Söllner 1989, S. 82f.
[72] Vgl. Wedekind; Ortner 1980, S. 31ff.
[73] Vgl. Ortner 2005a, S. 79.
[74] Vgl. Ortner 2005a, S. 83.

Die Aufgaben in der Phase Systementwurf lassen sich auch als ein „Entwickeln im Großen“ auffassen, bei der man sich mit der Architektur des Gesamtsystems, das aus Teilsystemen und Modulen besteht, beschäftigt[75].

Das Systemkonzept ist die Menge der Ergebnisdokumente der Phase des Systementwurfs. Sie liegt – wie schon die Ergebnisse der Voruntersuchung und des Fachentwurfs – der weiteren Entwicklungsarbeit nach dem MPVM zugrunde, weshalb das Entwickeln von Anwendungssystemen von Schienmann auch zutreffend als ein Prozess der Transformation und Konstruktion sprachlicher Ausdrücke aufgefasst wird[76].

V4 – Implementierung

In der Phase Implementierung geht es um das schrittweise Ausformulieren der einzelnen Programmkomponenten (Module). Hierbei kann man auch von einem „Programmieren im Kleinen“ [77] sprechen. Je nach gewähltem Anwendungssystemtyp wird nun das Systemkonzept in der jeweiligen Implementierungssprache (Programmiertechnik) umgesetzt und die physischen Daten-, Programm- oder Komponenten-Strukturen erzeugt. Diese Phase der Implementierung kann wiederum in sieben, zum Teil orthogonale, Teilaufgaben zerlegt werden[78]: Programmierung, physische Datenorganisation, Implementierung von Komponenten, Implementierung von Konnektoren, Implementierung von Transaktionsprogrammen, Implementierung von Benutzungsoberflächen, Implementierung von Architekturen. Die Tätigkeit der Implementierung gleicht einer Übersetzung von Diagrammsprachen zu maschinenverwertbarem (maschinenlesbarem) Code.

Als Ergebnisse der Phase Implementierung fallen vollständige Softwaresysteme oder aber die (Software-)Komponenten eines Anwendungssystems an.

V5 – Konfigurierung

In der Phase der Konfigurierung werden Komponenten (z. B. die Ergebnisse der Phase Implementierung) unterschiedlicher Kategorien gemäß den „(Bau-)Plänen“ aus dem Systementwurf zu einem Anwendungssystem verbunden (integriert). Dieses Verbinden gelingt mittels Konnektoren, wie bspw. Verbindungssprachen, die man auch „glue“ (dt. Klebstoff) nennt oder Frameworks, die eine Art Gestell für die Komponen-

[75] Vgl. Schienmann 1997, S. 25.
[76] Vgl. Schienmann 1997, S. 11.
[77] Vgl. Schienmann 1997, S. 25.
[78] Vgl. Ortner 2005a, S. 84.

ten eines Anwendungssystems abgeben. Mit Hilfe solcher Konnektoren können komplexe, mehr oder weniger lose Verbindungen zwischen den Komponenten hergestellt werden[79]. Die Komponenten lassen sich mit Hilfe von Komponentenkatalogen übersichtlich verwalten. Im Idealfall läuft die komponentenorientierte Anwendungsentwicklung wie folgt ab: *„Die Komponenten werden aus einem Katalog entnommen und dann nach einem Plan (z. B. einer Syntax oder Grammatik, auf der Basis eines Frameworks oder mit Hilfe von Stücklisten und Konstruktionsplänen) zusammengesetzt.*[80]*“*

Eine Software-Konfiguration ist eine vollständig zusammengefasste Menge von Entwicklungsergebnissen, die in ihrer Wirkungsweise und ihren Schnittstellen optimal aufeinander abgestimmt sind und gemeinsam eine vorgegebene Aufgabe erfüllen[81].

Die Anwendung, die nun bereits über eine reine Software-Lösung hinausgehen kann, ist das Ergebnis der Phase Konfigurierung.

V6 – Stabilisierung

In der Phase Stabilisierung ist die Binnenstruktur des Ganzen (sprachlicher und nichtsprachlicher Teil einer Anwendungssystem-Lösung) von Bedeutung. Hier werden Aufgaben zur Planung, Organisation und Arbeitsvorbereitung behandelt, um die Anwendung(en) geschickt in die Organisation (Prozesse und Aufgabenträger) einzubetten und mit ihr zu einem ergonomischen System zu integrieren. Primäraufgabe ist die Steigerung der Effizienz des Gesamtsystems[82]. Beim Aufbau des ergonomischen Systems, das auch soziotechnisches System genannt werden kann, wird auf die Aussagenbasis des Fachentwurfs zurückgegriffen. Dabei ist ein häufiges Wechseln zwischen der Organisationsmodellierung[83], einer evtl. Simulation[84] und der Benutzungsplanung und -steuerung des Gesamtsystems auf der einen Seite (Vorgangstyp Stabilisierung) und der notwendigen Wissensbeschaffung (Wissensrekonstruktion) für die Anwender auf der anderen Seite des MPVM (Vorgangstyp Fachentwurf) der Regelfall. Das Testen der (fachlichen) Korrektheit des Gesamtsystems mit Testfällen bzw. Test-Szenarien

---

79 Vgl. Ortner 2005a, S. 116.
80 Ortner 2005a, S. 115.
81 Vgl. Heinrich 2002, S. 277.
82 Vgl. Ortner 2005a, S. 131.
83 Vgl. Kosiol 1972.
84 Vgl. Wedekind et al. 1998.

gehört ebenfalls in diese Phase wie eine evtl. durchzuführende Pilotphase. Die Dokumentation des Anwendungssystems sollte am Ende dieser Phase abgeschlossen sein.

Ergebnis dieser Phase ist ein freigegebenes, ausgetestetes Anwendungssystem und seine komplette Beschreibung (z. B. in einem Repositorium-System).

V7 – Gebrauch

Die Phase Gebrauch umfasst das Aufgabenspektrum, wie es allgemein dem Informationsmanagement zugeschrieben wird. Aus aufgabenorientierter Sicht versteht man unter Informationsmanagement *„die Planung (z. B. des Betriebs), die Organisation und Distribution (von Ressourcen), die Hilfeleistung bei der Nutzung, die Abrechnung und Kalkulation sowie die Führung (mit Steuerung und Kontrolle) der anwender- und rechnerunterstützten Informationsverarbeitung (Einsatz von Informations- und Kommunikationstechnologie) in einem Unternehmen.*[85] *“*.

Im Gebrauch können zeitlich gesehen die fünf Phasen Entwicklung, Betrieb, Nutzung, Wartung und Beseitigung (Dekonstruktion) des Anwendungssystems bzw. der Anwendung auftreten. Im Rahmen des End-User Computing[86] spielen Aspekte des Usability Engineerings[87] eine große Rolle, in dem schrittweise im Gebrauch immer wieder über die Verbesserung der Mensch-Maschinen-Interaktion reflektiert wird. Gerade in dieser Phase des Gebrauchs können wieder neue Mängel festgestellt werden, so dass der Lebenszyklus der Anwendungssystementwicklung erneut im Sinne einer schrittweisen Verbesserung durchlaufen werden kann.

Zwischen die Vorgangstypen Stabilisierung und Gebrauch kann der Vorgang „Einführung“ eingeschoben werden. Zentrale Aufgaben der Einführung sind Schulungen der Anwender und der Umgang mit Altsystemen sowie evtl. notwendige Einführungsstrategien (Migration, Integration, Stichtagsumstellung etc.)[88], wobei die Anwenderschulungen auch schon in einer früheren Phase, z. B. Stabilisierung erfolgen können.

Begleitend zu den sequentiellen Vorgangstypen (V1 – V7) finden während des Anwendungssystem-Lebenszyklus permanent die Vorgänge Projektmanagement, Qualitätsmanagement und Betreuung der Anwender und Entwickler statt.

---

[85] Ortner 2005a, S. 153-154.
[86] Vgl. Scheidl 2006, S. 563.
[87] Vgl. Nielsen 2003.
[88] Vgl. Ortner 2005a, S. 151.

Die Anwendungsentwicklung ist ein Prozess der (Re-)Konstruktion und Transformation sprachlicher Ausdrücke[89]. Die Aufteilung der Entwicklungsphase in einen methodenneutralen Fachentwurf und einen methodenspezifischen Systementwurf bietet einen langfristigen Vorteil bei der Wiederverwendung des Fachkonzepts bspw. bei einem Technologiewechsel.

2.3.2 Lebenszyklus der Komponenten eines Anwendungssystems

Während der Phase „Gebrauch“ (siehe Abschnitt 2.3.1, V7) eines flexiblen komponentenbasierten Anwendungssystems kann ein weiterer Lebenszyklus identifiziert werden: Der Lebenszyklus der (Software-)Komponenten eines aus Komponenten zusammengesetzten Anwendungssystems. Dieser Lebenszyklus kann grundsätzlich in die vier Phasen: Herstellung, Vermarktung, Gebrauch und Abbau eingeteilt werden. Repositoriumbasierte Werkzeuge (engl. Tools) werden zur Unterstützung des Lebenszyklus der vom Anwendungssystem verwendeten Komponenten eingesetzt[90]. Abbildung 7 zeigt die Werkzeuge in Abhängigkeit zu den jeweiligen Phasen.

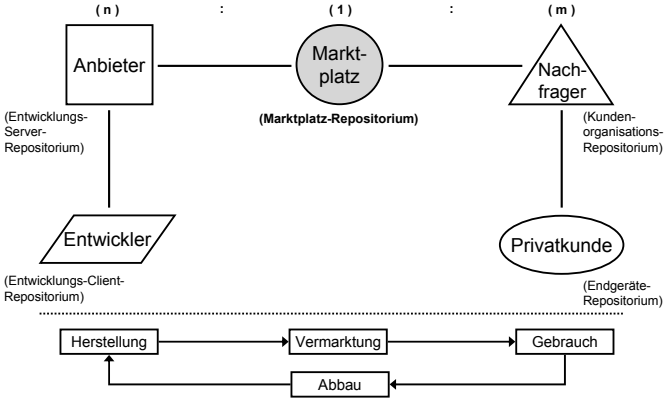

Abbildung 7: Werkzeuge und Phasen des Anwendungsmanagements[91]

In der Phase der Herstellung von Anwendungssystemen kommen dabei zwei Werkzeuge zum Einsatz. Der Entwickler wird durch ein Entwicklungs-Client-Repositorium bei der Entwicklung von Anwendungssystemen unterstützt. Dieses Repositorium kann an einem Entwicklungs-Server-Repositorium angeschlossen sein, das (Software-)Komponenten oder Module von Software-Anbietern bereitstellt. In einem sol-

[89] Vgl. Schienmann 1997.
[90] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 368-369.
[91] In Anlehnung an Grollius; Lonthoff; Ortner 2006a, S. 369.

chen Server-Repositorium kann sich der Entwickler z. B. mittels Komponentenkatalogen nach benötigten (Software-)Komponenten erkundigen.

In der Phase Vermarktung von Anwendungssystemen dienen elektronische Marktplätze als Marktplatz-Repositorien. Sie ermöglichen das Vermitteln (Match-Making[92]) von Anbieter und Nachfrager. Über solche Marktplätze können (Software-)Komponenten bedarfsgerecht bereitgestellt werden (siehe Abschnitt 4.2.3 und Kapitel 5). Für den Marktplatzbetreiber gibt es unterschiedliche Möglichkeiten wirtschaftlich von seiner Intermediärsfunktion zu profitieren (siehe Abschnitt 4.3).

In der Phase Gebrauch von Anwendungssystemen stehen zwei Werkzeuge zur Verfügung. Der (Privat-)Kunde wird durch ein Endgeräte-Repositorium im Gebrauch von Anwendungskomponenten (internes Anwendungsmanagement, siehe Abschnitt 2.5.1) unterstützt. Dieses Client-Repositorium kann an ein Kundenorganisations-Repositorium auf Nachfragerseite angeschlossen werden.

In der Phase Abbau von Anwendungssystemen kann auf das Marktplatz-Repositorium zurückgegriffen werden, um z. B. eine (Software-)Komponente „zurückzugeben", bspw. in Form der Beendigung eines Laufzeitvertrags (Beendigung der Lizenz bzw. der Geschäftsbeziehung). Das Marktplatz-Repositorium sorgt dann dafür, dass die „zurückgegebene" (Software-)Komponente rückstandsfrei aus dem Anwendungssystem entfernt wird. Dabei kann dieser Vorgang entweder auf dem Endgerät selbst oder in der Laufzeitumgebung der Serveranwendung stattfinden.

2.4 Der Management-Begriff in der Anwendungsinformatik

Funktional betrachtet beschreibt der Begriff „Management" die Gesamtheit der üblichen Tätigkeiten zur Führung bzw. Verwaltung von Organisationen. Unter diesem Aspekt fällt der Begriff in den Bereich der Unternehmensführung[93].

Der Begriff des „(allgemeinen) Managements" im betrieblichen Umfeld wird von Ulrich und Fluri funktional mit den Hauptaufgaben: *„Bestimmung der Unternehmenspo-*

[92] Hümmer 2004.
[93] Vgl. Drucker 2005, S. 95.

litik, Unternehmensplanung und Kontrolle, Organisation und Führung, sowie Kaderförderung[94]*"* (Führungskräftenachwuchs) zusammengefasst.

Wild definiert Management *„als die Verarbeitung von Informationen und ihre Verwendung zur zielorientierten Steuerung von Menschen und Prozessen.*[95]*"* Leinweber et al.[96] fassen die Aufgaben des Informationsmanagement mit Planung, Durchführung und Abwicklung zusammen. Ortner gibt in seinem Beitrag „Informationsmanagement – Wie es entstand, was es ist und wohin es sich entwickelt" eine Herleitung des Informationsmanagements und identifiziert die wesentlichen Aufgaben als das Management der Informationsverarbeitung, das Management der Ressource Information und der Verwaltung der Informationen über die organisationelle Informationsverarbeitung (Information Resources Dictionary System)[97]. Das klassische Informationsmanagement beschäftigt sich mit der Planung, Organisation und Führung der in sich verzahnten Lebenszyklen der Ressourcen Daten, Anwendungen, Technologie und Anwender unter Einbezug des aus der Sicht der Informationsverarbeitung relevanten Umfeldes des Unternehmens.

Drucker[98] fasst den Management-Begriff sehr weit und stellt ihn in jegliches Umfeld, bei dem etwas geführt, also geplant und gesteuert wird. Diese Auffassung aufgreifend wird in der vorliegenden Arbeit „Management" neutral definiert als die Gesamtheit der Aufgaben zur Planung, Steuerung und Kontrolle[99] während des gesamten Ressourcen-Lebenszyklus in Bezug auf einen Gegenstand.

Innerhalb der Anwendungsinformatik lassen sich mehrere Gegenstände identifizieren. So erfreut sich der Management-Begriff in der Anwendungsinformatik einer sehr hohen Beliebtheit. Neben dem Informationsmanagement, werden Teilbereiche und andere Gegenstandsbereiche einem Management unterzogen. Man findet bspw. Arbeiten zu Datenmanagement[100], Technologiemanagement[101], Terminologiemanagement[102], Wis-

94 Ulrich; Fluri 1995, S. 40.
95 Wild 1971.
96 Vgl. Leinweber; Andresen 1992, S. 24f.
97 Vgl. Ortner 1991a.
98 Vgl. Drucker 2005, S. 97.
99 Vgl. Krcmar 2000, S. 19.
100 Vgl. Gillenson 1982.
101 Vgl. Ortner 2005a, S. 164.
102 Vgl. Hellmuth 1997.

sensmanagement[103], Workflow-Management[104], Human Resource Management[105], Architektur-Management[106] etc. Alle diese, das Anwendungssystem betreffende Aufgaben lassen sich im Anwendungsmanagement integrieren (siehe Abschnitt 2.5). Diese möglichst nahtlose Integration ermöglicht eine höhere Qualität von Anwendungssystemlösungen[107].

2.5 Anwendungsmanagement

Vom Begriff her betrachtet ist Anwendungsmanagement ein spezielles Management. Will man die im vorangehenden Abschnitt getroffenen Definitionen bzw. Aufgaben des Managements in den Bereich des Anwendungsmanagements übertragen, so lässt sich das Management auf Anwendungen beziehen. Damit ist der hier verwendete Anwendungsmanagement-Begriff umfassender als z. B. seine Bedeutung im ARIS-Umfeld, in dem Anwendungsmanagement als Steuerung des Einsatzes von Anwendungssystemen zur Build-Time aufgefasst wird[108]. Anwendungsmanagement beschränkt sich auch nicht auf das Management der Entwicklung von Anwendungen, sondern schließt ebenso das Management der Ausführung der Anwendungen mit ein – auch im Kontext der Anwender. Somit soll Anwendungsmanagement nicht als Fortführung der Entwicklung in Form einer Wartungsphase betrachtet werden. Anwendungsmanagement bezieht sich also auf die Planung, Steuerung und Kontrolle des gesamten Lebenszyklus von Anwendungen[109]. Das Anwendungsmanagement besteht aus Aufgaben zum Aufbau, Betrieb (Einsatz) und Abbau (Dekonstruktion) von Software-Anwendungslösungen zur Rechnerunterstützung der Prozesse in einem Handlungsbereich.

Werden omnipräsente Anwendungssysteme aus verschiedenen heterogenen Elementen aufgebaut und wiederum in dynamische und ebenso heterogene Umgebungen eingebunden, dann muss das Zusammenwirken ihrer Elemente zur zweckgerichteten Erfüllung einer Aufgabe eines Nutzers geplant, gesteuert und im Gebrauch überwacht werden. Diese Aufgaben werden im Anwendungsmanagement zusammengefasst. Dabei können die unterschiedlichen Aufgaben des Anwendungsmanagements sowohl von

[103] Vgl. Bodendorf 2006, Kapitel 4.
[104] Vgl. Jablonski 1997.
[105] Vgl. Clark 1993.
[106] Vgl. Reussner; Hasselbring 2006, Kapitel 3.
[107] Vgl. Pagé Peter 1992, S. 100.
[108] Vgl. Wollnik 1988.
[109] Vgl. Ortner 1991a, S. 324.

Menschen wahrgenommen werden, als auch teilweise automatisiert, durch Rechner unterstützt, durchgeführt werden.

Planung

Mit Planung ist im Wesentlichen die Soll-Situation des Systemdesigns angesprochen. Zu diesem Zweck werden zuerst die Anforderungen der Nutzer herausgearbeitet. Dann werden die Bedingungen des Anwendungssystems erfasst, wie bspw. Ressourcenbeschränkungen der eingesetzten Hardware oder Aspekte der Sicherheit und Verfügbarkeit. Im Rahmen des Anwendungsmanagements sollte entschieden werden, in welcher Konfiguration und mit welchen Einstellungen eine Anwendung für einen speziellen Fall zu installieren ist. Kriterien können hierbei die Ressourcen des Endgerätes sein, auf dem die Anwendung genutzt werden soll. So würde im Rahmen eines automatisierten Anwendungsmanagements evtl. nur eine Minimal-Installation auf einem sehr ressourcenbeschränkten Endgerät erfolgen, während auf einem modernen PC eine vollständige Installation durchaus Sinn macht. Ein anderes Beispiel für Planung ist die Anpassung einer Benutzungsoberfläche an die Größe des zur Verfügung stehenden Bildschirms. Die komponentenorientierte Planung hat die Modularisierung eines Anwendungssystems zum Gegenstand. So wird bei der Planung über die Granularität der einzusetzenden (Software-)Komponenten bestimmt.

Steuerung

Steuerung bedeutet die (möglichst automatische, zumindest rechnerunterstützte) Veranlassung einer Aktion, welche die Ausführung von Anwendungen betrifft. Die Steuerung hat einen direkten Einfluss auf das Verhalten bestimmter Anwendungen. Das Anwendungsmanagement kann nicht nur eine beobachtende Funktion übernehmen, sondern muss aktiv die Ausführung von Anwendungen lenken können. Als Beispiel kann das Einrichten eines neuen Netzwerkteilnehmers mit Hilfe des Anwendungsmanagements verstanden werden. Der neue Teilnehmer muss in die existierende Struktur eingegliedert werden, sowohl auf Ebene der Hardware (Netzwerke und Engeräte), als auch auf Ebene der vorgegebenen Anwendungen. So kann es nötig sein, ein vorgegebenes Kommunikationssystem oder Adressverwaltungssystem mit diesem neuen Netzteilnehmer zu verbinden. Im Fall der komponentenorientierten Anwendungssysteme heißt Steuerung aber insbesondere die aktive Auswahl der jeweils optimalen Komponente innerhalb des Anwendungsmanagements.

Kontrolle
Kontrolle bezeichnet den Teilbereich des Anwendungsmanagements, der die Ist-Situation der Anwendungen im Betrieb (Gebrauch) zur Steuerung überprüft. Dies beinhaltet die Sicherstellung gewünschter Ziele, wie z. B. der Sicherheit oder einer gewünschten maximalen Datendurchsatzrate in einem Netzwerk. Nur durch das kontinuierliche Beobachten der Anwendungen (z. B. durch Überwachung festgelegter Zustände) kann das Anwendungsmanagement seinen gewünschten Einfluss geltend machen. So dürfen bspw. bei sicherheitsrelevanten Bereichen Anwendungen oder Nutzer nur kontrolliert Zugriff auf bestimmte Teilbereiche erlangen.

Bei komponentenbasierten Anwendungen umfasst Anwendungsmanagement die Aufgaben der Einführung von Anwendungen (Installation, Bereitstellung), der Konfiguration der Anwendung innerhalb der auszuführenden Umgebung, der Überwachung und Steuerung der ordnungsgemäßen Ausführung der Anwendung sowie die geplante Dekonstruktion (Abbau) der Anwendung.

Anwendungsmanagement ist von dem klassischen Änderungsmanagement (Change Management[110]) zu unterscheiden. Änderungsmanagement bedeutet in der Regel die Planung und Steuerung der kontrollierten Veränderung einer Anwendung durch Anpassung der Software oder Austausch von Komponenten unter Berücksichtigung der Abhängigkeiten von und der Auswirkungen auf das System. Weiterhin ist Anwendungsmanagement vom Konfigurationsmanagement (z. B. Component Configuration Management[111]) zu unterscheiden. Ein Hauptziel des Konfigurationsmanagements ist die vollständige Dokumentation und dadurch volle Transparenz eines Erzeugnisses (z. B. Software oder Komponente) sicherzustellen[112]. Der in dieser Arbeit verwendete Anwendungsmanagement-Begriff ist umfassend und schließt die anderen hier genannten Managementansätze (siehe auch Abschnitt 2.4) mit ein.

Im Falle komponentenbasierter Anwendungssysteme gibt es den umfassenderen Begriff des „Anwendungssystem-Managements“, bei dem ganze Anwendungssysteme Gegenstand der Planung, Steuerung und Kontrolle des Aufbaus (Konfiguration), Einsatzes und Abbaus von Anwendungssystemen zur Erfüllung eines Handlungszwecks mit Rechnerunterstützung sind. Hierbei werden der Aufbau und der Betrieb von Soft-

[110] Vgl. Versteegen; Salomon; Heinold 2001.
[111] Vgl. Crnkovic; Larrson 2000.
[112] Vgl. Versteegen 2003, S. 6.

ware-Produkten bspw. für die Unterstützung des täglichen Lebens von Nutzern behandelt. Dabei werden die Software-Produkte entsprechend den individuellen Bedürfnissen der Nutzer zusammengesetzt und abgestimmt.

Aufbauend auf einem solchen Anwendungssystem-Management stellt ein Anwendungssystem-Management-System, wie in Grollius et al.[113] vorgestellt, eine Weiterentwicklung der klassischen Anwendungsentwicklungsumgebung[114] (AEU, siehe Abbildung 8) insbesondere für komponentenbasierte Anwendungssysteme dar. Die repositoriumbasierte AEU unterstützt das Management der Anwendungssysteme während ihres gesamten Lebenszyklus.

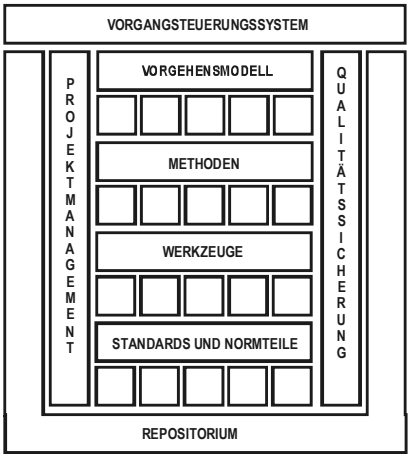

Abbildung 8: Anwendungsentwicklungsumgebung[115]

Die Komponenten der Anwendungsentwicklungsumgebung, wie Vorgehensmodell, Methoden, Werkzeuge, Normteile und Standards, Qualitätssicherung, Projektmanagement, Repositorium und Vorgangssteuerungssystem werden in Abbildung 8 idealtypisch zu einer anpassbaren Entwicklungsumgebung für Anwendungssysteme zusammengesetzt.

Dabei umfasst das Anwendungssystem-Management-System die Aufgaben: Beschaffung geeigneter Komponenten, Aufbau eines Anwendungssystems, Konfiguration des

[113] Grollius et al. 2005.
[114] Vgl. Ortner 1991b, S. 421.
[115] In Anlehnung an Ortner 2005a, S. 203.

Gesamtsystems, Überwachung und Steuerung des Gebrauchs des Anwendungssystems, Dekonstruktion von Komponenten des Anwendungssystems, organisatorische Veränderungen und das Hardwaremanagement (Gerätemanagement etc.). Es ist also ein System, das Anwendungsmanagement bereitstellt, erweitert um das Management der Organisation und der Hardware.

Zur Unterstützung dieser Aufgaben bietet ein Anwendungssystem-Management-System geeignete Werkzeuge an. Dabei können unterschiedliche Grade der Automatisierung realisiert werden. Es können Werkzeuge zur Unterstützung einer strukturierten Vorgehensweise bereitgestellt werden. Das langfristige Ziel ist die integrierte automatische Erfüllung der Aufgaben des Anwendungsmanagements. Für flexible komponentenbasierte Anwendungssysteme ist eine solche vollautomatische Unterstützung denkbar[116].

Orthogonal zu der Unterscheidung zwischen Anwendungsmanagement und Anwendungssystem-Management, wird in Abschnitt 2.5.2 zwischen internem und externem Anwendungsmanagement unterschieden.

2.5.1 Aspekte des Anwendungsmanagements in mobilen verteilten Systemen

Einfache Betriebssysteme, wie bspw. Windows, sind ein gutes Beispiel dafür, wie Aufgaben des Anwendungsmanagements größtenteils automatisch erfüllt werden. Betriebssysteme sind verantwortlich für eine funktionierende Infrastruktur für Anwendungen, indem sie deren Ausführung planen, steuern (Entscheidungen, wenn auch einfacher Art, bezüglich deren Ausführung treffen) und überwachen (z. B. in Bezug auf Verzeichniszugriffe oder Status der Anwendung). Ein Betriebssystem kann all diese Aufgaben gut wahrnehmen, da es sich in einer lokalen und aus Sicht des Anwendungsmanagements in einer überschaubaren Umgebung befindet.

Komplexer wurde die Situation durch die zunehmende Vernetzung der Computer. Anfangs übernahmen meistens noch einfache Betriebssysteme die gesamte Verwaltung, die dabei nur um Methoden und Werkzeuge der Kommunikation ergänzt wurden. Zunehmend wurden jedoch Ansätze entwickelt, das Anwendungsmanagement auch auf das Netzwerk selbst zu übertragen. Dies war ein Entwicklungsschritt analog zu der Entwicklung computerübergreifender Anwendungen. Dieses Anwendungsmanagement innerhalb eines Betriebssystems befasst sich hauptsächlich mit der computer-

[116] Vgl. Grollius et al. 2005, S. 213f.

übergreifenden Kommunikation in einem solchen Netzwerk. Während eine ursprüngliche Aufgabe des Anwendungsmanagements die Planung und Verteilung der lokalen Ressourcen eines Computers beinhaltet, so gilt dies nun für die Ressourcen des Netzwerkes (i. S. von Cyberspace oder Grid-Computing). Wurde zuvor der Rechenaufwand einer Anwendung kontrolliert, so muss nun auch die Netzlast, die eine Anwendung hervorruft, mit einbezogen werden.

Daraus resultierend befasst sich das Anwendungsmanagement auch mit dem Management von Netzwerken. Meist wird diese Funktion des Anwendungsmanagements von Systemen einer zentralen Middleware wahrgenommen (Kommunikations- bzw. Netzwerkmanagement[117]). Aber noch immer existieren viele Anwendungen jeweils nur auf einzelnen Endgeräten und kommunizieren mit anderen Anwendungen, indem sie Daten austauschen.

Mobile verteilte Anwendungen (z. B. auch Software-Agenten[118]) nutzen jedoch die Vorteile der inzwischen hoch entwickelten Vernetzung und platzieren sich im Netzwerk quasi selbst. Sie sind in der Lage flexibler zu reagieren, eine konzentrierte und integrierte Datenhaltung zu ermöglichen und dadurch die Integration der Anwendungen stärker zu fördern. Solche Anwendungen finden sich jedoch noch immer nur selten, zunehmend jedoch in Unternehmens- oder Forschungsnetzwerken. Aufgrund dieser Situation sind auch die aktuellen Fähigkeiten eines Anwendungsmanagements in den Netzwerken meist im Hinblick auf eine Verwaltung der Datenströme in Netzwerken optimiert. Das Anwendungsmanagement in Netzwerken hat sich zu einer Kombination des Anwendungsmanagements auf den einzelnen Endgeräten (gemäß eines Betriebssystems) und des Anwendungsmanagements für das Netzwerk bspw. in Form einer Middleware herausgebildet.

Über die letzten Jahre erweiterten sich die Einsatzmöglichkeiten mobiler verteilter Systeme, so dass man heute in der Lage ist, mit hoch entwickelten mobilen Netzwerken zu arbeiten. Im Rahmen dieser fortschreitenden Entwicklung entsteht zunehmend ein Bedarf an Anwendungsmanagementfunktionalitäten für das mobile Umfeld. Das Anwendungsmanagement in der Form, wie sie aktuell in klassischen Netzwerken genutzt wird, lässt sich aufgrund der in Abschnitt 3.1.3.4 vorgestellten Eigenschaften des mobilen Umfeldes jedoch nur sehr begrenzt in mobilen Anwendungssystemen einset-

[117] Vgl. Häckelmann; Petzold; Strahringer 2000, S. 207ff.
[118] Vgl. Mutschler; Specht 2004, S. 73-78.

zen. Aufgrund dieser Eigenschaften werden vier Problemfelder bzw. Aspekte des Anwendungsmanagements eingeführt.

Die drahtlose Kommunikation und die Mobilität führen zu einer dynamischen Netzwerkarchitektur, welche sich z. B. in stark variierenden Bandbreiten bis hin zu Netzwerkabbrüchen äußert. Das Anwendungsmanagement in klassischen Netzwerken hat sich mit diesen Problemen bisher nur wenig auseinandergesetzt und die Lösung solcher Aufgaben meist ganz an die Protokolle bzw. deren Adapter abgegeben. Aus diesen Gründen ergibt sich für das Anwendungsmanagement ein weit reichendes Problemfeld unter dem Aspekt der Kommunikation.

Der zweite Aspekt ergibt sich aus der Heterogenität mobiler Netzwerke, die vor allem dadurch entsteht, dass nicht mehr ein Netzwerkadministrator entscheidet, welche Geräte an das Netzwerk angeschlossen werden, sondern ein Nutzer mit seinem mobilen Endgerät einfach den Empfangsbereich eines mobilen Netzes betritt. Die Überwindung der Produkt-, Plattform- und Technologieheterogenität mobiler Endgeräte führt zu der Notwendigkeit einer geeigneten Datenhaltung und einem geeigneten Datenaustausch. Die ausgetauschten Daten unterschiedlichster Art müssen für alle potenziellen Teilnehmer des Netzwerkes interpretierbar und damit verständlich sein. Dies ist ein für das Anwendungsmanagement entscheidender Punkt, da Anwendungen nur funktionieren, wenn die Daten genutzt werden können. Aus diesem Grund liegt ein besonderes Gewicht auf einer hoch entwickelten Datenmodellierung für mobile verteilte Systeme.

Aufgrund der Transportierbarkeit mobiler Endgeräte ergeben sich Einschränkungen der Benutzbarkeit mobiler Endgeräte. Dieses Problemfeld führt direkt zur Notwendigkeit der Betrachtung des Aspekts der Benutzbarkeit (Usability). Auch im Rahmen dieses Aspektes spielt die Heterogenität der Endgeräte eine wichtige Rolle, da das Anwendungsmanagement auch für eine effiziente Nutzbarkeit der Anwendungen auf unterschiedlichen Endgeräten, z. B. mit verschiedenen Bildschirmgrößen oder verschiedenen Möglichkeiten der Dateneingabe bzw. -ausgabe, verantwortlich ist.

Ein weiterer, für die Akzeptanz der Anwender wichtiger Aspekt ist die Sicherheit. Gerade in Verbindung mit der Nutzung drahtloser Netzwerktechnologien und lokaler Datenhaltung in mobilen Endgeräten besteht ein hoher Bedarf an Sicherheitslösungen, die durch das Anwendungsmanagement zu integrieren sind.

Diese vier oben kurz betrachteten Aspekte können nicht immer eindeutig voneinander getrennt werden, da sie jeweils verschiedene Dimensionen derselben Problematik, des Anwendungsmanagements in mobilen verteilten Systemen, darstellen. Im Folgenden werden diese vier Aspekte vertieft.

2.5.1.1 Kommunikation

Mit dem Begriff „Kommunikation“ wird im Allgemeinen ein Prozess bezeichnet, bei dem Informationen zwischen zwei Teilnehmern (im unidirektionalen Fall nur vom Sender zum Empfänger) nach bestimmten Regeln ausgetauscht werden. Der Kommunikationsbegriff wurde zunächst ausschließlich auf den Bereich der zwischenmenschlichen Interaktion eingeschränkt. Mit der Entwicklung des Computers wurde eine weitere Dimension der Kommunikation erschlossen, in der sich sowohl technische als auch sprachliche Elemente wiederfinden. Hierbei spielen sowohl der Austausch zwischen Mensch und Maschine, der hier dem Aspekt „Benutzbarkeit“ zugeordnet wird, als auch die Übertragung von Informationen innerhalb eines Netzwerks von kommunikationsfähigen Geräten eine wichtige Rolle.

Damit gilt es den Shannon’schen Informationsbegriff[119] und das darauf basierende Kommunikationsmodell um das sprachlogische Kommunikationsmodell[120] zu erweitern. Dazu muss eine Sprachschichtung mit Hilfe des Begriffspaars „Schema/Ausprägung“ eingeführt werden. Ein Schema kann über allgemeine Aussagen rekonstruiert werden. Eine singuläre Aussage führt zu einer Ausprägung (Instanz). Zwischen Schema und Ausprägung gibt es richtungsabhängige Verbindungen (Sprachhandlungen). Ausgehend von einem Schema kann eine Ausprägung geäußert werden. Ausgehend von einer Ausprägung kann etwas mit Hilfe des korrespondierenden Schemas verstanden werden. Diese Beziehungen werden in Abbildung 9 dargestellt.

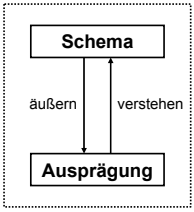

Abbildung 9: Schema-Ausprägung[121]

[119] Vgl. Shannon; Weaver 1949.
[120] Vgl. Lorenz 1992, S. 113.
[121] In Anlehnung an Ortner 2005a, S. 77.

Über die Sprachhandlungstypen „äußern“ und „verstehen“ läuft auch die Kommunikation zweier Sprachteilnehmer ab. Dabei erfolgt die eigentliche Kommunikation auf Basis von Ausprägungen, die zwischen den Kommunikationsteilnehmern ausgetauscht werden.

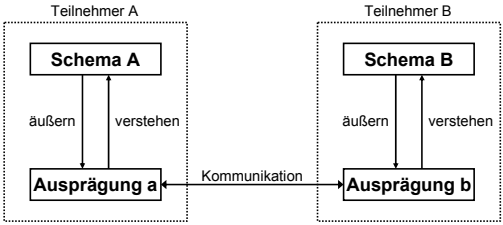

Abbildung 10: Sprachlogisches Kommunikationsmodell[122]

Dieses sprachlogische Kommunikationsmodell nach Lorenz[123] erweitert das Syntax-orientierte nachrichtentechnische Kommunikationsmodell um eine inhaltliche Modellierungsmöglichkeit. Lorenz beschreibt den in Abbildung 10 dargestellten Kommunikationszusammenhang wie folgt: *„Von demjenigen, für den es die Handlung aktiv gibt – der sie ‚tut’ – sagt man auch, dass er ein ‚token’ oder eine ‚Aktualisierung’, die Handlung als etwas Einzelnes (Singulares; single act) erzeugt habe; von derjenigen, für die es die Handlung passiv gibt – die sie ‚erleidet’ – sagt man hingegen, dass sie ein ‚type’ oder ein ‚Schema’, die Handlung als etwas Allgemeines (Universales; generic action) erkannt habe.[124]“*

Eine solche Kommunikation verläuft erst dann erfolgreich (im Sinne des Verstehens), wenn beide Kommunikationsteilnehmer jeweils über ein passendes Schema verfügen, über das die Ausprägungen verstanden werden. Nicht nur bei Kommunikationsprozessen von Menschen untereinander, auch zwischen Mensch und Maschine bzw. umgekehrt oder Maschine zu Maschine ist eine Verständigung ohne Übereinstimmung der Schemata der Kommunikationsteilnehmer nicht möglich. Dazu gilt es, bei der Modellierung bzw. der Organisation von Kommunikationsprozessen Heterogenitäten zu überwinden.

[122] In Anlehnung an Ortner 2005a, S. 156.
[123] Vgl. Lorenz 1992, S. 113.
[124] Lorenz 1992, S. 113.

Die möglichen Heterogenitäten lassen sich dabei in die folgenden Heterogenitätsarten einteilen[125]:

- *technische Heterogenität*
  Sie resultiert aus der Verwendung unterschiedlicher, inkompatibler Kommunikationsnetze (Hardware und Software).

- *syntaktische Heterogenität*
  Es werden bspw. unterschiedliche Formate für die Darstellung von Daten- und Programmschemata eingesetzt.

- *semantische Heterogenität*
  Dieselben Ausdrücke stehen (inhaltlich) für verschiedene Gegenstände oder dieselben Gegenstände werden von verschiedenen Ausdrücken referenziert.

- *pragmatische Heterogenität*
  Ausdrücke lösen bei den sie Interpretierenden (Mensch oder Maschine) unterschiedliche Handlungen oder Operationen aus.

- *taktische Heterogenität*
  Der Ablauf der Kommunikation (oder Datenverarbeitung) wird bei sonst einvernehmlichen Tatbeständen von den Kommunikationsteilnehmern (Mensch und/oder Maschine) unterschiedlich betrieben.

Alle Heterogenitätsarten lassen sich bei der Konstruktion der Kommunikationsprozesse über Format- und (Inhalts-)Standards, Protokolle oder allgemein über eine sogenannte „Zwischensprache“ und die Befolgung grundlegender Regeln beim Aufbau und Betrieb von Anwendungssystemen überwinden[126].

Dieser Kommunikationsbegriff schließt alle Anforderungen, Prozesse, Methoden und Technologien ein, die bei der Interaktion mehrerer Geräte auftreten können. Dabei können diese Geräte, von denen mindestens eines mobil sein muss, sowohl aus dem Bereich mobiler verteilter Systeme, als auch aus dem stationären Umfeld kommen. Die unterschiedlichen resultierenden Architekturen führen zu technischen und fachli-

[125] Vgl. Ortner 2004, S. 148.
[126] Vgl. Ortner; Wedekind 2003.

chen Problemstellungen, die im Rahmen des Anwendungsmanagements behandelt werden sollen. Dabei können zwei Problembereiche unterschieden werden: Vernetzung sowie Dienst- und Ressourcenverwaltung.

Vernetzung

Ein erster Teilaspekt konzentriert sich auf die physische Vernetzung der Geräte. Basierend auf der physischen Infrastruktur wird ein Anwendungsmanagement entwickelt, das eine reibungslose Interaktion zwischen den verteilten Anwendungen ermöglichen soll. Hierbei muss beachtet werden, dass die fehlerfreie Kommunikation der Endgeräte nicht Aufgabe des Anwendungsmanagements ist und nicht als gewährleistet vorausgesetzt werden kann, da drahtlose Netzwerke unzuverlässige Kommunikationsnetze sind. Diese spielt bei dem Entwurf des Anwendungsmanagements nur insofern eine Rolle, als dass es sich an die aus der physischen Infrastruktur resultierenden Anforderungen anpassen muss.

Um einen fehlerfreien und möglichst effizienten Betrieb der mobilen verteilten Anwendungen zu gewährleisten, muss das Anwendungsmanagement die sogenannte physische Umgebung berücksichtigen. Dazu gehören die unterschiedlichen Netzwerkstrukturen, Kommunikationsprotokolle, Übertragungsarten und -bandbreiten sowie sämtliche Hardware-Parameter, welche die Möglichkeiten der Rechnerbelastung ausmachen.

Bezüglich der Strukturen verteilter Systeme werden klassische, ad-hoc und nomadische Systeme unterschieden (siehe Abschnitt 3.1.3.4). Hinsichtlich der Übertragung können synchrone und asynchrone Kommunikationsarten unterschieden werden. Desweiteren ergibt sich die Frage, wo die anfallenden Daten abgelegt werden, auf dem Endgerät oder auf Ressourcen die durch das Netzwerk bereitgestellt werden. Hierbei werden Konzepte zur Offline- und Online-Datenhaltung bzw. hybriden Verfahren benötigt[127].

Dienst- und Ressourcenverwaltung

Ein weiterer Teilaspekt betrifft die Koordination der Interaktion verschiedener Komponenten. Komponenten, die hier Software- und Hardwarekomponenten sein können, interagieren miteinander, um Daten auszutauschen und Dienste anzubieten bzw. in Anspruch zu nehmen. Die Gewährleistung der Komponenteninteraktion zwischen den

[127] Vgl. Mutschler; Specht 2004, S. 4f.

mobilen verteilten Komponenten einerseits und der mobilen und stationären Komponenten andererseits ist ein Schwerpunkt des Anwendungsmanagements in mobilen verteilten Systemen.

Einer der wichtigsten Vorteile der mobilen drahtlosen Kommunikation ist die Möglichkeit, Dienste in Anspruch zu nehmen. Dabei sind Dienste bereitgestellte Funktionalitäten von Komponenten, die von anderen Endgeräten über ein Netzwerk bereitgestellt werden. Eine Kalenderkomponente, die auf einem stationären Server betrieben wird, kann durch ein externes mobiles Endgerät abgefragt oder auch von ihm selbst zum Teil ausgeführt werden. In diesem Zusammenhang muss festgestellt werden, welche Komponente eines Netzwerks einen gewissen Dienst anbieten kann und wie eine bestimmte Komponente einen neuen eigenen Dienst für andere Komponente zur Verfügung stellen kann. Hinsichtlich der Dienstverwaltung können zentralisierte oder dezentralisierte Systeme unterschieden werden, die das Vermitteln der Anfragen und Antworten von Diensten entsprechend regeln.

Eng mit der Verwaltung von Diensten verbunden ist das Ziel der optimalen Zuordnung von Ressourcen wie Energie, Rechenleistung, Speicher etc. innerhalb des Netzwerks. Für ressourcenaufwändige Prozesse (z. B. Bildbearbeitung) müssen Lösungen gefunden werden, um Aufgaben innerhalb des Netzwerkes zu verteilen, so dass einzelne Komponenten nicht überlastet werden. Mit Blick auf die Mobilität und Portabilität der hier betrachteten Endgeräte erscheint die Optimierung von Rechenleistung als ein interessanter und zukunftsträchtiger Aspekt. Der auf das Endgerät abgestimmte Umgang mit Ressourcen kann sich weiterhin positiv auf die Benutzbarkeit des Systems auswirken.

2.5.1.2 Datenmodellierung

Ein weiterer wichtiger Bestandteil der Interaktion verschiedener Komponenten innerhalb des Netzwerks ist der Datenaspekt. Daten aus verschiedenen Anwendungsgebieten werden auf mobilen oder stationären Geräten gespeichert und durch unterschiedliche beteiligte Komponenten angefragt bzw. zur Verfügung gestellt. Ein reibungsloser und effizienter Ablauf dieser Prozesse kann durch eine entsprechende Datenintegration erleichtert werden. Darauf basierend kann eine Integration verschiedener Anwendungen erfolgen, da durch die Datenmodellierung Interoperabilität geschaffen wird. Eine solche Integrationsbasis erleichtert die Realisierung eines integrierten Anwendungsmanagements.

Im mobilen Umfeld ist die Möglichkeit der Modellierung von Ortsabhängigkeiten besonders interessant. Durch die Hinzunahme des Ortsbezugs können lokale Komponenten identifiziert werden und ortsabhängige Abfragen sofort realisiert werden[128].

Aufgaben der Datenmodellierung, die sich als die Organisation der Daten durch Klärung der Fachbegriffe sowie der Zusammenhänge zwischen den Begriffen versteht, sind sowohl die Überwindung der Heterogenität zwischen Anwendungen und Systemen als auch die sprach- und sachgerechte Rekonstruktion von Wissen. Das durch ein übergreifendes konzeptionelles Schema organisierte Wissen kann dann zwischen den verschiedenen Anwendungen und Geräten übertragen werden[129].

Überwindung der Heterogenität

Die Anwendungsintegration basiert im Unternehmen auf einer gemeinsamen und anwendungsübergreifenden Datenmodellierung. Darüber hinaus wird durch externes Anwendungsmanagement versucht, unternehmensübergreifend die Heterogenität zu überwinden und dabei die Eigenschaften der mobilen Umgebung zu berücksichtigen.

Bei der Entwicklung von Standards müssen spezielle Einschränkungen des mobilen Umfeldes berücksichtigt werden. Besondere Beachtung kommt der ressourcen- und zugriffszeitoptimalen Darstellung von Daten zu. Dabei kann man z. B. eine Baumdarstellung mit einer objektorientierten Datenkapselung und einer XML-basierten Speicherung vergleichen. Lösungen dieses speziellen Problems des Anwendungsmanagements werden durch Werkzeuge, wie bspw. das Bayou-Speicherungssystem (siehe Abschnitt 3.5.1.7), das Coda-Dateisystem (siehe Abschnitt 3.5.1.6) sowie die Odyssey-Architektur (siehe Abschnitt 3.5.1.6) realisiert, die auf vergleichbaren Formaten basierende Datenkonsistenz und Homogenität zusichern.

Sprach- und sachgerechte Rekonstruktion

Wie bereits im sprachlogischen Kommunikationsmodell von Lorenz (siehe Abschnitt 2.5.1.1) verdeutlicht, basiert eine erfolgreiche Interaktion zwischen zwei Teilnehmern auf dem Austausch von Äußerungen, die durch vergleichbare Schemata erzeugt wurden. Durch Äußerungs- bzw. Verstehenshandlungen sind die Kommunikationsteilnehmer in der Lage, das eigene Schema (Wissen) in Form von Ausprägungen (Information) zu vermitteln bzw. zu aktualisieren.

---

[128] Vgl. Höpfner; Türker; König-Ries 2005, S. 36f und 41ff.
[129] Vgl. Ortner 1984a, S. 46.

Das Management verschiedener Anwendungen im mobilen Umfeld bietet ausgehend von diesem Modell weitere Chancen und Herausforderungen. Unterschiedliche mobile Endgeräte können eigens erzeugte Informationen an Geräte weitergeben, die über erforderliche Rechenkapazitäten oder Sensoren nicht verfügen. Eine derartige Erweiterung des Netzwerkes bringt Themen wie Wissensmanagement sowie die Vermarktung und insbesondere Erreichbarkeit von Wissen auch hier ins Spiel.

Zusammenfassend kann festgestellt werden, dass die Datenmodellierung eine wichtige Komponente des Anwendungsmanagements darstellt, da sie eng mit der primären Funktion des Anwendungsmanagements, der problemlosen Interaktion unterschiedlicher Anwendungen innerhalb des Netzwerks, verbunden ist.

2.5.1.3 Benutzbarkeit

Der Aspekt der Benutzbarkeit bezieht sich auf alle Elemente der Mensch-Computer Interaktion, wobei die Eigenschaften des mobilen Umfelds eine entscheidende Rolle spielen (siehe Abschnitt 3.1.3). Als besondere externe Eigenschaften werden solche Eigenschaften des Endgerätes bezeichnet, die durch den Benutzer unmittelbar wahrgenommen werden können, wie z. B. Bildschirmgröße und -auflösung, Anzahl und Größe der Tasten. Davon sind besondere interne Eigenschaften zu unterscheiden, welche die interne Ausstattung des Gerätes betreffen, wie z. B. Prozessorleistung und Speichergröße.

Besondere externe Eigenschaften

Ein wesentlicher Faktor in der Benutzbarkeit von mobilen Endgeräten spielt die Präsentation von Daten, die entscheidend von den externen Eigenschaften der Geräte abhängt. Dieses Problem tritt insbesondere dann auf, wenn sich das Gerät, auf dem die Anwendung ausgeführt wird, und dasjenige, auf dem die grafische Darstellung der Ergebnisse erfolgt, nicht identisch sind, so dass die externen Eigenschaften der Geräte nicht mehr von der Anwendung selbst berücksichtigt werden können. Es gehört zu den Aufgaben des Anwendungsmanagements, für eine intuitive und benutzerfreundliche Masken- und Grafikverarbeitung zu sorgen, die sogar mögliche Einschränkungen des Benutzers berücksichtigt. Die entsprechende Darstellung von Komponenten oder Daten, die über das Netzwerk vermittelt wurden, sollte gewährleistet werden. Weitere externe Eigenschaften können z. B. bei der Druckerverwaltung oder Multimediaaufbereitung auftreten.

Besondere interne Eigenschaften

Anders als im Desktop Computing stehen bei mobilen Endgeräten meist nur stark eingeschränkte Ressourcen zur Verfügung. Von besonderer Bedeutung sind dabei die Energiespeicherkapazitäten, die Rechenkapazitäten und der verfügbare Speicher. Das Anwendungsmanagement übernimmt in diesem Zusammenhang die Aufgabe, die Ressourcen eines Endgerätes möglichst optimal zu verwalten und darüber hinaus innerhalb des Netzwerks eine optimale Zuordnung zu erreichen. Besonders rechenintensive Prozesse könnten bspw. vorübergehend an andere leistungsfähigere Geräte übergeben werden. Durch Reduktion der Verbindungszeit mit dem Netzwerk können zusätzlich Verbindungskosten eingespart werden und der Energieverbrauch durch Abschaltung des Netzwerkmoduls gesenkt werden.

Die Sicherung der Benutzbarkeit ist neben der Gewährleistung von Kommunikation im weitesten Sinne eine weitere wichtige Aufgabe des Anwendungsmanagements, da sie am besten die aus Kundensicht entstehenden Vorteile der Vernetzung in mobilen verteilten Systemen verdeutlicht.

2.5.1.4 Sicherheit

Der bereits im Bereich von stationären verteilten Systemen überaus wichtige Sicherheitsaspekt, gewinnt vor dem Hintergrund des mobilen Umfelds noch mehr an Bedeutung. Durch die drahtlose Verbindung zwischen Geräten wird der unberechtigte Zugang zu den ausgetauschten Datenpaketen entscheidend erleichtert, so dass besondere Maßnahmen für die Sicherung der Verbindung erforderlich sind. Da mobile Endgeräte häufig unmittelbar aus einem Netzwerk in ein anderes wechseln können, können hier die Voraussetzung der fest verkabelten Geräte nicht mehr getroffen werden. Ein Dritter kann möglicherweise die Identität des Benutzers im ursprünglichen Netzwerk behalten oder diejenige im neuen Netzwerk annehmen.

Ein weiteres sicherheitsrelevantes Problem stellt die Mobilität der mobilen Endgeräte an sich dar. Diese meist handlichen Endgeräte können leichter entwendet oder gar verloren werden. Weist das Endgerät keine Zugriffsschutzmechanismen auf, ist es für Dritte sehr leicht an schutzwürdige Daten zu gelangen. Für solche Fälle gibt es erste Ansätze von Schutzmechanismen, die ein Fernauslösen eines vollständigen Hardware-Resets (Kill-Pill) ermöglichen[130].

[130] Vgl. Lonthoff 2007, S. 12.

Sicherheit lässt sich im Wesentlichen durch die fünf Schutzziele Vertraulichkeit, Authentizität, Integrität, Nichtabstreitbarkeit und Verfügbarkeit beschreiben. Vertraulichkeit bedeutet, dass die Daten nur von dem Empfänger, nicht aber von unberechtigten Dritten eingesehen werden können. Durch Authentifikation wird die Identität überprüft und somit sichergestellt, dass sensible Daten auch wirklich nur an den beabsichtigten Empfänger gesendet werden, bzw. wirklich vom Absender stammen. Integrität bedeutet, dass Daten nicht verändert werden, sei es durch Manipulation oder Übertragungsfehler. Durch Nichtabstreitbarkeit wird eine nachvollziehbare verbindliche Handlung gewährleistet. Mit Verfügbarkeit wird der Systemzugriff gewährleistet[131].

Vertraulichkeit, Authentizität und Integrität können mit Hilfe kryptographischer Verfahren gewährleistet werden. Über die Mittel der Kryptografie hinaus gehen allerdings Fragen des Datenschutzes in Bezug auf die ausgetauschten Daten. Hierbei muss geklärt werden, ob der Sender hinreichend über den Umfang und der anschließenden Verwendung der übertragenen Daten informiert ist. Auch wenn die technischen Mittel der Geheimhaltung von Daten gegenüber Dritten bestehen, bleibt die Frage nach den Daten offen, die überhaupt einer Verschlüsselung unterzogen werden müssen. Auch dies liegt in der Verantwortung des Anwendungsmanagements.

2.5.2 Abgrenzung zwischen internem und externem Anwendungsmanagement

Komponentenbasierte Anwendungssysteme und Anwendungen sind dadurch gekennzeichnet, dass sie schon bei der Entwicklung arbeitsteilig und komponentenorientiert modelliert und konstruiert werden. Sie werden aus vorgefertigten, ggf. aber auch individuell hergestellten, wieder verwendbaren (Software-)Komponenten aufgebaut, die in einem Komponentenkatalog (oder Repositorium, das als ein elektronischer Marktplatz implementiert sein kann) dokumentiert sind und aus diesem zum Einsatz in einem Anwendungssystem entnommen werden. Die Verbindung der (Software-)Komponenten zu einem Anwendungssystem sollte lose gekoppelt sein (z. B. über ein Workflow-Management-System), so dass Änderungen an dem Anwendungssystem leicht und ohne hohen Codierungsaufwand vollzogen werden können.

Die Abgrenzung von internem und externem Anwendungsmanagement entstand auf Grundlage vieler Diskussionen am Fachgebiet. Eine tiefgehende Motivation zur Tren-

[131] Vgl. Lonthoff; Ortner 2007, S. 6.

nung dieser beiden Bereiche in Analogie zur klassischen Trennung von Betriebs- und Geschäftsprozessen in der Betriebswirtschaftslehre findet sich in Grollius[132].

Im Kontext der komponentenorientierten Anwendungssystementwicklung kann internes und externes Anwendungsmanagement über die Herkunft der in einem Anwendungssystem verwendeten (Software-)Komponenten unterschieden werden.

Alle entwicklungs- und konfigurationsrelevanten Aufgaben werden dem internen Anwendungsmanagement zugeschrieben. Das externe Anwendungsmanagement hat die Aufgabe, (Software-)Komponenten von außen z. B. von einem (Software-)Komponenten-Marktplatz oder als Dienst bspw. in Form von Web Services in den Anwendungssystem-Lebenszyklus zu integrieren.

Internes Anwendungsmanagement
Das interne Anwendungsmanagement kann als eine Aufgabe des Unternehmens, das ein Anwendungssystem einsetzt, aufgefasst werden. Hierbei umfasst das interne Anwendungsmanagement die lebenszyklusabhängigen Aufgaben Planung, (interne) Bereitstellung, Konfigurierung, Gebrauch und Abbau.

Als erste Phase im Lebenszyklus einer Anwendung oder eines Anwendungssystems erfolgt die Planung (Spezifikation). In dieser Phase werden Pläne für den Aufbau der Anwendung aus Komponenten im Hinblick auf die auszuführenden Prozesse aufgestellt und die benötigten (Software-)Komponenten fachlich beschrieben. Diese Beschreibung dient dann zur Suche nach geeigneten (Software-)Komponenten in einem unternehmensinternen Komponentenkatalog (Repositorium), aus denen die tatsächlich zu verwendenden ausgewählt werden können. An die Spezifikation schließt sich die Bereitstellung der benötigten (Software-)Komponenten an. In der Phase Konfiguration werden bereits im Unternehmen vorhandene (Software-)Komponenten sowie durch das externe Anwendungsmanagement zeit- und sachgerecht bereitgestellte externe (Software-)Komponenten planvoll zu einem Anwendungssystem verbunden.

Während des Gebrauchs einer Anwendung steuert das interne Anwendungsmanagement den Einsatz und überwacht ihn. Unzuverlässige, nicht performante oder fehlerhafte (Software-)Komponenten gilt es, aus dem Anwendungssystem auszulösen und durch besser geeignete zu ersetzen. Ggf. werden (Software-)Komponenten auch gegen

[132] Vgl. Grollius 2004.

neuere oder adäquatere ausgetauscht. Nach Erfüllung des Gebrauchszwecks wird die Anwendung kontrolliert abgebaut (Dekonstruktion) und die (Software-)Komponenten aus der Anwendung bzw. dem Anwendungssystem gelöst.

Bei reinen Software-Anwendungen kann dieser Prozess selbsttätig durch ein sogenanntes Anwendungssystem-Management-System (AMS) dynamisch ausgeführt werden. Eine detaillierte Darstellung der Teilaufgaben des Anwendungsmanagements und einer werkzeugseitigen Unterstützung durch ein Anwendungssystem-Management-System findet sich in dem Beitrag von Berbner et al.[133].

Externes Anwendungsmanagement

Das externes Anwendungsmanagement[134] ist dafür verantwortlich, dass (Software-)Komponenten von außen für ein Anwendungssystem-Management-System bereitgestellt werden. Als mögliche Formen dieser Komponentenbereitstellung können Repositorien, auf deren Basis bspw. die Komponentenkataloge realisiert sind, dienen. Zum anderen können Entwicklungs-(Server/Client)-Repositorien bereitgestellt werden, um bestehende (Software-)Komponenten bei der Entwicklung von Anwendungssystemen zu beschaffen (einzubinden). Eine weitere Form ist ein elektronischer Marktplatz, der unternehmensübergreifend als Ausprägung des B2B-Commerce bzw. konsumentenorientiert als Ausprägung des B2C-Commerce gestaltet ist[135].

Über die reine Bereitstellung von (Software-)Komponenten i. S. eines Komponententransfers hinaus kann auch die Funktionalität einer (Software-)Komponente als Dienst „von außen“ – außerhalb eines Unternehmens oder dem Besitz an Komponenten eines Anwenders – einem Endgerät oder einer anderen Komponente angeboten werden. Dazu kann bspw. ein externes Anwendungsmanagement eine Laufzeitumgebung instanzieren, die eine benötigte externe (Software-)Komponente ausführt und das Ergebnis übermittelt. Hierbei ist der Unterschied zwischen logischer und physischer Wiederverwendung von Komponenten zu beachten. Wird die Komponente auf ein Endgerät transferiert, so handelt es sich um eine logische Wiederverwendung, denn die Komponente wird dabei auf dem Endgerät repliziert. Es wird also nur die gleiche Komponente wieder verwendet. Läuft die Komponente im Netzwerk und das Endgerät nutzt ihren

[133] Berbner et al. 2005.
[134] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 368.
[135] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 369.

Dienst, dann wird die im Netzwerk physisch vorhandene (d. h. also dieselbe) Komponente verwendet. Man spricht hierbei von physischer Wiederverwendung[136].

Im weiteren Verlauf der Arbeit bezieht sich das Anwendungsmanagement im Wesentlichen auf das externe Anwendungsmanagement.

[136] Vgl. Ortner 2005a, S. 117.

3 Architektur- und (Inhalts-)Standards zur Entwicklung und zum Betrieb integrierter Anwendungssysteme

Um einen Einblick in die Vielfalt der zur Zeit existierenden Systeme und Ansätze im Bereich des Anwendungsmanagements geben zu können, werden zunächst solche vorgestellt, die im weitesten Sinne Funktionen des Anwendungsmanagements erfüllen bzw. entsprechende Teilaufgaben übernehmen können. Dabei wird bei dieser Aufzählung verwandter Arbeiten kein Anspruch auf Vollständigkeit erhoben, sondern es soll ein Überblick über die wesentlichen Ansätze geschaffen werden.

Hierbei lassen sich die Arbeiten in die drei Kategorien Basissystemarchitekturen, Anwendungsarchitekturen und (Inhalts-)Standards und Technologien der Implementierung für das Lebenszyklus-Management von Anwendungssystemen einteilen (siehe Abbildung 11)[137]. Darüber hinaus werden auch Anwendungssystemarchitekturen betrachtet, die das Ergebnis des Entwicklungsprozesses nach Abbildung 11 darstellen.

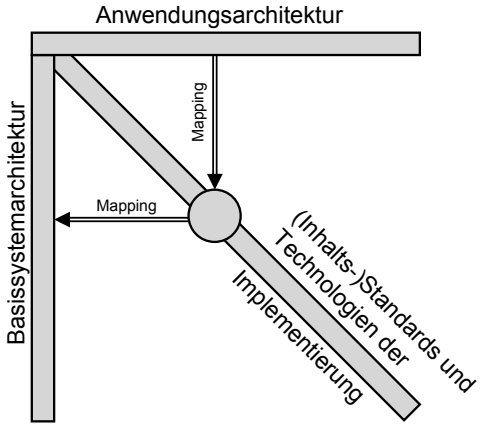

Abbildung 11: Dimensionen und Mittel des Lebenszyklus-Managements von Anwendungssystemen[138]

[137] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 365.
[138] In Anlehnung an Jablonski 2004, S. 10.

3.1 Anforderungen

Hinsichtlich der Anforderungen an Anwendungssysteme ist in der System- und Anwendungsinformatik der letzten 30 Jahre eine Entwicklung von der Forderung nach Integration über die Verteilung hin zur Mobilität festzustellen[139]. Dabei konkatenieren die Anforderungen bezüglich des Lebenszyklus-Managements, wie in Abbildung 12 dargestellt. Ein mobiles verteiltes System setzt also Integration ebenso voraus wie ein verteiltes (nicht mobiles) System.

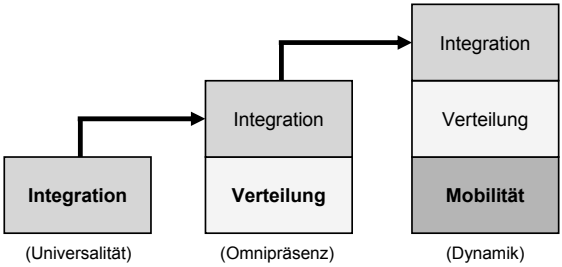

Abbildung 12: Konkatenation der Anforderungen[140]

3.1.1 Universalität

Eine Grundvoraussetzung für das Zusammenwirken heterogener Elemente in einem System ist ihre Integration. Bei dieser Aufgabe wird Universalität erreicht, indem die dafür einzusetzenden Mittel überall gültig sind. Als Mittel der Integration dienen bspw. globale Standards, Normen, Protokolle und entsprechende Übersetzer zur Überwindung von Heterogenitäten (siehe Abschnitt 3.5.2.10)[141].

3.1.2 Omnipräsenz

Vertrat man kürzlich noch die Auffassung, Ubiquitous Computing[142] sei ein Teilbereich des Mobile Computing[143], so sind die Erkenntnisse heute gerade dem entgegengesetzt (siehe auch Abschnitt 3.1.3.3). Mobile Computing umfasst die technischen Grundlagen mobiler Systeme mit dem Fokus auf mobile Endgeräte. Bei Ubiquitous Computing stehen integrierte und eingebettete Systeme im Mittelpunkt der Betrachtung, die dennoch vom Benutzer jederzeit und überall genutzt werden können. An-

---

[139] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 366.
[140] In Anlehnung an Grollius; Lonthoff; Ortner 2006a, S. 366.
[141] Vgl. Ortner 2005a, S. 257.
[142] Vgl. Weiser 1991.
[143] Vgl. Mutschler; Specht 2004.

wendungen hingegen sind omnipräsent, wenn sie auf ubiquitärer Hardware lauffähig sind. Die „Allgegenwart“ ubiquitärer Hardware bedingt ebenso omnipräsente Anwendungen, da die Hardware alleine meist nutzlos ist.

Ein „Gegenstand“ (z. B. ein Ding oder ein Geschehnis) ist omnipräsent, wenn er mindestens eine der folgenden Eigenschaften aufweist[144]:

- *Er ist überall vorhanden.*
  Der Gegenstand ist an jedem Ort verfügbar bzw. ist in jedem Gegenstand vorhanden. Hier geht es um den Hauptaspekt der Verteilung.

- *Er ist überall gültig.*
  Der Gegenstand besitzt zeit- und ortsunabhängig Gültigkeit, z. B. weltweit einheitlich gültige Standards. In diesem Zusammenhang ist genauer von Integration die Rede.

- *Er kann/könnte überall auftreten (Potenzial).*
  Wenn die notwendigen Voraussetzungen geschaffen wurden, kann sich der „Gegenstand“ (z. B. eine Komponente oder ein Dienst) an jeden Ort begeben oder könnte potenziell von dort genutzt werden. Dieser Aspekt der Omnipräsenz kann hier auf Mobilität zurückgeführt werden.

Omnipräsenz setzt also Integration und Verteilung voraus. Von dieser Unterscheidung lassen sich Konzepte des Pervasive Computing[145,146] abgrenzen, die nur auf den Aspekt des überall Durchdringens, also einer Omnipräsenz i. S. von „in allen Gegenständen vorhanden“ abzielen.

3.1.3 Dynamik

Die Dynamik eines Systems lässt sich nach innen (Anpassbarkeit) und nach außen (Migration) beschreiben. Auf jeden Fall beinhaltet die Eigenschaft „Dynamik“ den Aspekt der Zeit und des Sich-bewegen-könnens (Potenzial). Immer wenn sich in einem Zeitraum etwas ändern kann, ist es dynamisch (veränderbar). Dynamik wird dabei als ein „Können“ (z. B. in Form verfügbaren Wissens) von Systemen, sich neuen Um-

[144] Vgl. Grollius; Lonthoff; Ortner 2006a, S. 367.
[145] Vgl. Roth 2005.
[146] Vgl. Hansmann et al. 2001.

ständen (z. B. Prozessen) möglichst selbstständig bzw. selbsttätig anzupassen, definiert.

Eine Anwendung, die sich aus Komponenten zusammensetzt und sich je nach Anforderung möglicherweise von selbst (selbsttätig) in geeigneter Weise zusammenschließen kann, ist dynamisch, da die tatsächliche Konfiguration im Laufe der Zeit veränderbar ist.

Zum Erreichen dieser Dynamik ist Mobilität eine Grundanforderung. Intuitiv bedeutet Mobilität – auch geistiger Art – Sich-Bewegen-Können. Moderne Suchmaschinen erzielen bei der Suche nach dem Begriff „mobil" bzw. des englischen Äquivalents „mobility" mehrere Millionen Treffer (ca. 20 Mio. Treffer bei www.google.de, Stichwort „mobil", Stand 04.12.06; ca. 77 Mio. Treffer bei www.google.de, Stichwort „mobility", Stand 04.12.06).

Der Begriff „Mobilität" ist kontextabhängig. Ein Arzt versteht unter Mobilität etwas anderes als bspw. ein Soziologe, ein Autoverkäufer oder ein Philosoph. Zum anderen ist, wie schon intuitiv erkannt, der Kernpunkt von Mobilität die Beweglichkeit – nur aus verschiedenen Sichtweisen.

Vom Wortstamm her betrachtet entstammt „Mobilität" dem lateinischen Wort mobilitas[147], was in der Grundbedeutung soviel wie Beweglichkeit und Schnelligkeit heißt. Ging es früher meist um Mobilität im Militär („Truppen mobilisieren"), wird der Begriff heute in den unterschiedlichsten Zusammenhängen benutzt.

Rotach, Hoppler und Mötteli[148] schlagen eine differenzierte Einteilung nach Faktormobilität, Standortmobilität, Bevölkerungsmobilität, soziale Mobilität, räumliche (Bevölkerungs-)Mobilität und Pendelmobilität vor.

Eine weitere Aufteilung wurde von Cerwenka[149] vorgeschlagen. In Abbildung 13 ist diese Aufteilung, jedoch um „andere" Mobilitätsarten erweitert, abgebildet.

[147] Vgl. Stowasser; Petschenig; Skutsch 1994.
[148] Vgl. Rotach; Hoppler; Mötteli 1986.
[149] Cerwenka et al. 2000.

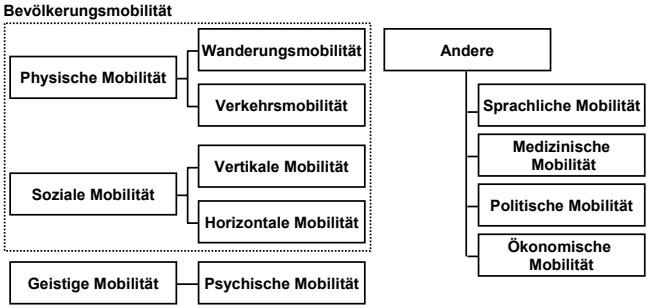

Abbildung 13: Übersicht der verschiedenen Mobilitätsbereiche[150]

Aus Abbildung 13 kann man entnehmen, dass physische und soziale Mobilität häufig unter dem Begriff Bevölkerungsmobilität zusammengefasst werden.

Physische Mobilität

Im Gesundheitswesen wird mit „Mobilität" die Beweglichkeit des menschlichen Bewegungsapparates bezeichnet. In diesem Zusammenhang spricht man auch von Mobilisieren, wenn bspw. Blockaden der Gelenke gelöst werden.

Eine weitere Betrachtung, die in eine ähnliche Richtung geht, ist die der menschlichen Bewegung generell, also nicht nur die Funktionsweise der Muskeln und Gelenke, sondern vielmehr deren (Aus-)Wirkung. Hiermit ist die Fähigkeit gemeint, sich von einem Punkt zu einem anderen bewegen zu können. Es handelt sich hier um ein menschliches Grundbedürfnis, ja sogar um einen *„unverzichtbaren Teil menschlicher Existenz*[151]*"*. Holzapfel sieht Mobilität *„als Wort für die ständige Abwesenheit von einem Ort, die ständige Sucht nach anderem*[152]*"*.

Dieses Grundbedürfnis zeigt sich bereits im frühesten Kindesalter. Sobald der Nachwuchs weiß, wie man robbt, ist er das erste Mal aktiv mobil und versucht ständig, „neue Gebiete" zu erreichen und seine Fähigkeit auszubauen. Mit zunehmendem Alter erhöht sich die Mobilität des jungen Menschen. Er lernt zu krabbeln, zu laufen, Fahrrad zu fahren. Später kommen die inzwischen alltäglichen Fortbewegungsmittel, wie Auto, Bahn, Schiff und Flugzeug hinzu.

---

[150] In Anlehnung an Cerwenka et al. 2000.
[151] Holzapfel 2001.
[152] Holzapfel 2001.

Die meisten Menschen assoziieren mit physischer Mobilität das Auto als einen Gegenstand der Mobilität ermöglicht. Dies trifft in zweifacher Hinsicht zu[153]: So unterscheidet Vogt zwei bedeutende „Revolutionen der Mobilität" im Laufe der letzten 200 Jahre. Zunächst erfolgte mit der Industrialisierung und besonders der Erfindung der Dampfmaschine *„eine Phase der Mobilität, nachdem es Jahrtausendelang nur im Schritttempo voran ging[154]"*. Die Dampfmaschine erlaubte es, neue Geschwindigkeiten zu erreichen, egal ob zu Lande oder auf dem Wasser. Man konnte sich nun 10-20fach schneller bewegen und so erstmals effizient größere Strecken zurücklegen (siehe Abbildung 14). Die Massenmotorisierung in den westlichen Ländern im Laufe der 1960er Jahre stellt die zweite Welle der „Mobilitätsrevolution" dar. Aufgrund dieser Entwicklung wurde vielen Menschen erstmals die Möglichkeit der Mobilität geboten zumindest in den Industrienationen.

Das Automobil trägt seinen Namen zu Recht, da es zum einen die Mobilität generell und zum anderen auf Grund der hohen Verbreitung die Mobilität eines jeden Einzelnen beträchtlich erhöht. Ohne das Auto wäre die Zivilisation, so wie sie heute ist, nicht möglich (in positiver und negativer Hinsicht).

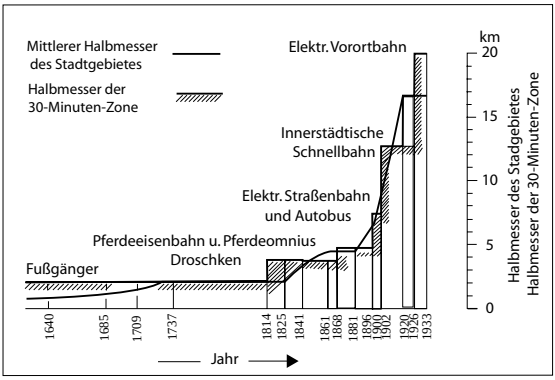

Abbildung 14: Stadtausdehnung und Verkehrsmittel[155]

Diese Entwicklungsstufen der physischen Mobilität können orthogonal dazu in Wanderungsmobilität und Verkehrsmobilität (siehe Abbildung 13) unterteilt werden. *„Unter Wanderungsmobilität fasst man räumliche Bewegungen von Haushalten zusam-*

[153] Vgl. Vogt 2002.
[154] Vogt 2002.
[155] Lehner 1966.

men, mit denen ein dauerhafter Wechsel der Wohnung bzw. des Wohnortes verbunden ist. Zirkuläre Mobilität (Verkehrsmobilität) bezieht sich demgegenüber auf die täglich wiederkehrenden Ortsveränderungen der Haushalte und ihrer Mitglieder.[156]*"*

Mit der Telekommunikation kündigt sich heute bereits ein dritter Sprung in der Entwicklung der Mobilität an[157]. So kann die „Reisezeit" bei einer Videokonferenz entfernter Teilnehmer durch die Lichtgeschwindigkeit der ausgetauschten Signale auf nahezu Null reduziert werden. Dem Menschen wird zeichenbedingt, semiotisch oder digital (Signale), die Möglichkeit geboten, an jedem beliebigen Ort intellektuell zu erscheinen. In diesem Kontext entstand der Begriff „virtuelle Mobilität" (siehe Abschnitt 3.1.3.1)[158].

Soziale Mobilität

Soziale Mobilität beschreibt die Bewegungen von Individuen innerhalb der Berufspositionen bzw. der sozialen Schichtung einer Gesellschaft (zwischen zwei Zeitpunkten gemessen). Im Vergleich zur physischen Mobilität ist dies eine ganz andere Form der Mobilität. Allerdings kann man bei näherer Betrachtung auch feststellen, dass soziale und physische Mobilität in Wechselwirkung stehen. Ein Wechsel der Berufsposition ist z. B. nicht selten mit einer Veränderung der geographischen (physischen) Mobilität verbunden und umgekehrt.

Einen Schritt weiter geht die Definition von sozialer Mobilität im Brockhaus: *„Beweglichkeit in Bezug auf den Beruf, die soziale Stellung, den Wohnsitz. Positionenwechsel, die keine Statusänderung einschließen, werden als horizontale Mobilität, soziale Auf- und Abstiegsprozesse als vertikale Mobilität bezeichnet.* [159]*"* Hier kommt eine zweite Dimension ins Spiel, denn es wird zwischen vertikaler und horizontaler sozialer Mobilität unterschieden.

Geistige Mobilität

Entsprechend des semantischen Bewegungsaspektes beschreibt hier die „Mobilität" geistige Beweglichkeitspotenziale: Gedankliche Flexibilität, intellektuelle Beweglichkeit und Kreativität. *Geistige Mobilität* kann als die Fähigkeit eines Menschen verstanden werden, über den Einsatz verfügbarer Schemata zu reflektieren, neue Schema-

[156] Hautzinger; Tassaux-Becker; Pfeiffer 1996.
[157] Vgl. Vogt 2002.
[158] Vgl. Lonthoff 2007, S. 3.
[159] Zwahr 2000.

ta zu erwerben und bestehende Schemata, wenn dies geboten erscheint, zu modifizieren (Reflexion)[160].

Andere Mobilitätsarten

Sprachliche Mobilität ist die Überwindung von Barrieren, wie sie üblicherweise vorherrschen. Schneider spricht sogar von *„Welten, in denen sich Individuen befinden und die es zu durchbrechen und verbinden gilt[161]“*. Diese Sprachbarrieren sind ganz allgemein zu verstehen, da sie nicht nur die unterschiedlichen Sprachen der Welt meinen, wie man intuitiv vermuten könnte. Vielmehr geht es ganz abstrakt um das „Verstehen“ selbst: Wenn ein Gesprächsteilnehmer etwas äußert, kann es passieren, dass er von einem anderen Gesprächsteilnehmer nicht verstanden wird (selbst dann, wenn beide die gleiche Sprache beherrschen und verwenden). Es besteht also im Allgemeinen nicht nur ein syntaktisches Verständigungsproblem (verschiedene Sprachen), sondern auch ein semantisches (Bedeutung und pragmatisches (Handlung) des Gesagten).

Medizinische Mobilität betrachtet die Beweglichkeit i. e. S. (die aktive Beweglichkeit in einem Körpergelenk)[162]. Zum anderen beschäftigt man sich aber auch mit dem Handlungsbedarf der Medizin in der mobilen Gesellschaft[163].

Politische Mobilität entsteht dadurch, dass sich die Menschen immer weniger an bestimmte Parteien, Verbände, Institutionen und Führer binden, sondern sie entscheiden sich stets von neuem und für kürzere Zeit, ob und wo sie sich engagieren.

Ökonomische Mobilität beschäftigt sich mit der Beweglichkeit und dem Wechsel von Gütern. Man kann leicht eine Einteilung von Gütern in verschiedene, nach Beweglichkeit sortierte Klassen vornehmen. Immobilien i. A. sind bspw. nicht beweglich, während die meisten Konsumgüter mehr oder weniger mobil sind.

3.1.3.1 Virtuelle Mobilität

Neben diesen klassischen Definitionen von Mobilität entstand ein neuer Begriff der „virtuellen Mobilität“. Dabei wird der Begriff „virtuell“ im Zusammenhang mit der Informationstechnologie häufig verwendet, ohne dass ein Bewusstsein vorliegt, welche

[160] Vgl. Lonthoff 2004, S. 453.
[161] Schneider 1992.
[162] Vgl. Reiche 2003.
[163] Vgl. Westhoff 2000.

Bedeutung dieser Begriff hat. Zunächst lassen sich nach Wedekind zwei Formen der Virtualität unterscheiden[164]: abstrakte und semiotische Virtualität.

Die abstrakte Virtualität basiert auf dem Abstraktionsprinzip als zentralem Konzept zur Virtualisierung. Die semiotische Virtualität kommt zustande durch den *„Vollzug einer Zeichenhandlung beziehungsweise durch das Herausstellen von Ergebnissen*[165]*“*. Diese beiden Formen der Virtualität lassen sich in einem Begriffstetraeder darstellen (siehe Abbildung 15).

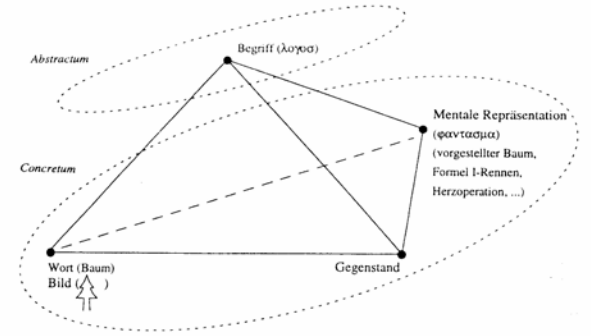

Abbildung 15: Abstrakte und semiotische Virtualität im Begriffstetraeder[166]

Abstrakte Virtualität

Die abstrakte Virtualität ist demzufolge auf der Spitze des Tetraeders einzuordnen, im Bereich des Abstrakten. Sie bewegt sich in der Welt abstrakter Begriffe, die zunächst nur Sammlungen von Eigenschaften sind, die ohne konkrete Auslegung nicht begreifbar gemacht werden können. Dies geschieht erst durch die Zuordnung von Begriffswörtern zu konkreten Gegenständen.

Man verwendet das Prinzip der abstrakten Virtualität zur Entwicklung von Systemen, die in einer Vielzahl unterschiedlicher Einsatzbereiche die gleiche Funktionalität für den Anwender bereitstellen sollen. Ein einfaches Anwendungsbeispiel für abstrakte Virtualität ist der sogenannte virtuelle Speicher für den Einsatz in Betriebssystemen[167]. Dieser abstrahiert von den physikalisch vorhandenen Speichermedien eines Computersystems. Mit einer speziellen Verwaltung werden virtuelle Adressen in physische bzw.

[164] Vgl. Wedekind 1999, S. 104.
[165] Vgl. Wedekind 1999, S. 105.
[166] Wedekind 1999, S. 106.
[167] Vgl. Wilkes 1970.

physische in virtuelle umgesetzt. Sinn und Zweck eines virtuellen Speichers ist es, dem Rechner einen Speicher anzubieten, der größer ist, als der eigentlich physikalisch vorhandene Hauptspeicher. Dies wird erreicht, indem andere Speichermedien, wie Festplattenspeicher, mit Hilfe eines Speichermanagements zum Hauptspeicher hinzugenommen werden. Ein abstrakter Speicher ist virtuell, wenn er invariant bezüglich seiner physischen Speicherung adressierbar ist. So bleibt die Adressierung insbesondere unverändert, wenn die Auslagerungsdatei von einem Sekundärspeichermedium auf ein anderes verschoben wird.

Durch Virtualisierung in Form von Abstraktion wird auf diese Weise eine gewisse Unabhängigkeit von konkreten Anwendungsszenarien und Einsatzumgebungen geschaffen. Insbesondere kann man diese Form der Virtualität zur Bewältigung von Heterogenitätsproblemen einsetzen.

Semiotische Virtualität

Unter semiotischer Virtualität versteht man allgemein die digitale Repräsentation von Dingen oder Zeichenhandlungen. Der Vollzug einer Zeichenhandlung ist dabei die Darstellung von etwas Konkretem in Wort oder Schrift. Semiotische Virtualität bezieht sich nicht direkt auf abstrakte Begriffe, sondern basiert auf dem Raum der konkreten Objekte, auch derjenigen Objekte, die für Menschen nicht direkt wahrnehmbar sind, wie bspw. Nanostrukturen.

Virtualität in der Mobilität

Die modernen Formen der Kommunikationstechnologien machen es möglich, künstliche – virtuelle – Umfelder zu schaffen, in denen sich Menschen bewegen bzw. handeln können. Bei dieser Kommunikationsform steht in der Regel die digitale Repräsentation von konkreten Objekten, wie bspw. Merkmale der Kommunikationsteilnehmer (SMS-Textbotschaften ersetzen das menschliche Gespräch mit Mimik, Ton und Dialog[168]) im Vordergrund. Mobilität in einer derart abstrahierten Form des Informationsaustausches kann die zuvor beschriebene, physische Mobilität vollständig ersetzen und z. B. durch Telearbeit in einigen Bereichen sogar überflüssig machen. So kann die „Reisezeit“ bei der Nutzung einer Videokonferenz entfernter Teilnehmer durch die Lichtgeschwindigkeit der ausgetauschten Signale auf nahezu Null reduziert werden[169]. Dem Menschen wird zeichenbedingt, semiotisch oder digital (Signale) die Möglichkeit

[168] Vgl. Bell; Gray 1997, S. 5.
[169] Vgl. Vogt 2002.

geboten, an jedem beliebigen Ort intellektuell zu erscheinen. Mobile Anwendungen ermöglichen eine derart funktionale Simulation kommunikativer Nähe, sodass man berechtigterweise davon sprechen kann, dass moderne Telekommunikations-, Netzwerktechnologien und mobile Anwendungen die Welt im reinsten Wortsinne viel eher zusammenbringen als dies allein durch physische Massenmobilität möglich wäre.

Gleichzeitig prägt der aktuell stattfindende Aufbruch bislang dominierender räumlicher Zugangs- und Erreichbarkeitsbarrieren in Richtung allumfassender „Überallität" auch semantisch neue Begrifflichkeiten: Ubiquitous und Pervasive Computing und die Omnipräsenz von Anwendungssystemen. *„Information Anytime, Anywhere*[170]*"* – jede Art der Information[171] ist zu jeder Zeit an jedem Ort präsent[172].

Die Mobilität geistig arbeitender Menschen wird durch virtuelle Mobilität unterstützt. Der Wissensarbeiter kann dabei unabhängig von seinem Aufenthaltsort überall seine Dienste in Form von eigener intellektueller Leistung erbringen. Diese Leistung wird dann über Netzwerke an die Stelle weitergeleitet, von der die Leistung angefordert wurde.

Neben dem durch virtuelle Mobilität entsprechend veränderten Verständnis von Raum, Nähe und Distanz, tritt – bedingt unter anderem durch netztechnischen Fortschritt, gerade in den Bereichen, für die heute die räumliche Dimension des „Überalls" bereits als selbstverständlich angenommen wird – das Zeitkriterium immer stärker in den Vordergrund.

Auch wenn die Vision einer vollkommen virtuellen Welt[173] dargestellt, ein wenig zu futuristisch erscheint, ist eine rasant zunehmende Abbildung der realen physischen Welt in einen virtuellen Cyberspace deutlich zu sehen (z. B. Augmented Reality[174] und Virtual Reality[175], wie z. B. Second Life[176]). In dem mobilen, ortsbezogenen Abenteuerspiel „Mobile Hunters" wird eine Vernetzung zwischen virtueller Welt und physi-

[170] Gates 1999.
[171] Im Bezug auf Medien bezieht sich dies auch auf (informative) Inhalte des Kommunikationsmediums.
[172] Vgl. Mattern 1999.
[173] Wie bspw. in dem Film „The Matrix"™ (http://whatisthematrix.warnerbros.com).
[174] Vgl. Wiedenmaier 2004.
[175] Vgl. Kim 2005.
[176] Secondlife.com.

scher Welt durch die Verbindung beider Realitäten erreicht[177]. Technologien im Bereich drahtloser Kommunikationsnetze, Lokalisierungsmethoden und der Einsatz von (mobilen) Datenbanken zielen darauf ab, bestehende reale Dinge, Personen und Sachverhalte digital abzubilden.

Dieser Zusammenhang zwischen physischer und digitaler (geistiger) Welt wird von Ortner verallgemeinert als Workflow-Management-Anwendung aufgefasst und in folgender Abbildung 16 abstrakt darstellt[178].

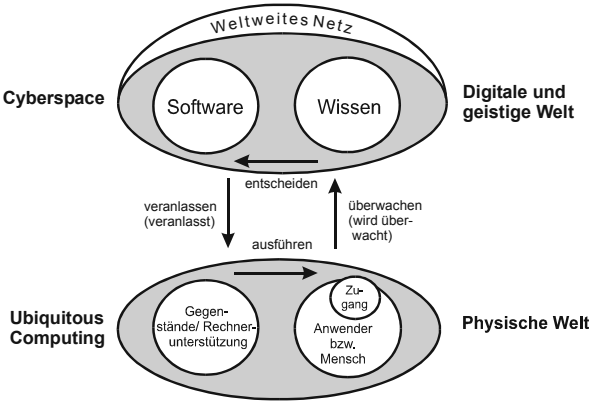

Abbildung 16: Zusammenhang physische und digitale Welt[179]

In einem Anwendungsbereich werden die Objekte der physischen Welt aus dem Cyberspace heraus, dem Workflow-Management-System oder der digitalen (geistigen) Welt, zu Vorgängen (Handlungen) veranlasst und mit erzielten Ergebnissen (physische Welt) aus der digitalen (geistigen) Welt heraus überwacht.

Im Rahmen von Ubiquitous Computing werden immer mehr Gegenstände mit Informations- und Kommunikationsinfrastruktur ausgestattet, so dass Prozesse bzw. Geschehen der physischen Welt der Steuerung und Kontrolle durch den Cyberspace unterliegen. Die Kontrolle kann durch entsprechende Software hergestellt aber auch manipuliert werden.

---

[177] Vgl. Lonthoff; Leiber 2005.
[178] Vgl. Ortner 2005a, S. 129f.
[179] In Anlehnung an Ortner 2005a.

3.1.3.2 Mobile Kommunikationsarten

Im mobilen Umfeld ist nicht immer ganz klar, worauf sich die Mobilität bezieht. Zunächst soll der Unterschied von „drahtlos“ und „mobil“ aufgezeigt werden, um mobile Kommunikation von Mobilkommunikation unterscheiden zu können. Weiterhin muss die Reichweite der Mobilität bei der Unterscheidung von Kommunikationsarten mit berücksichtigt werden.

Drahtlos heißt nicht zwangsläufig mobil, wie man leicht an folgenden Beispielen in Tabelle 1 entnehmen kann:

	Nicht-mobile Kommunikation	Mobile Kommunikation
Drahtgebundene Kommunikation	PC über Netzwerkkabel mit dem Netzwerk verbunden	Notebook (z. B. in einem Hotel) per Modem mit dem Netzwerk verbunden
Drahtlose Kommunikation	PC über WLAN mit dem Netzwerk verbunden	Notebook per Mobiltelefon mit dem Netzwerk verbunden

Tabelle 1: Beispiele zum Unterschied von drahtloser und mobiler Kommunikation[180]

Mobilität liegt also nur vor, wenn man sich von überall mit einem Netzwerk (z. B. GSM-Netz oder Internet) ohne Einschränkungen verbinden kann. Ein Benutzer ist also mobil, wenn er sich innerhalb eines bestimmten Systems oder einer bestimmten Zelle frei bewegen kann.

Eine weitere Flexibilität wird dadurch erworben, dass sich der Benutzer unabhängig von bestimmten Zellen bewegen kann ohne gleichzeitig Funktionalität einzubüßen. Dadurch wird der Begriff „Mobilität“ untergliedert in „Intra-Zell-Mobilität“ und „Inter-Zell-Mobilität“.

Ein Computer, der über WLAN mit dem Netzwerk verbunden ist, kann z. B. nur innerhalb seiner lokalen Umgebung bewegt werden. Diese Eigenschaft macht ihn flexibel. Er kann an unterschiedlichen Orten innerhalb der Zellreichweite betrieben werden, ohne dass die Verbindung verloren geht. Bewegt man den Computer hingegen an einen Ort außerhalb der Reichweite des WLAN-Senders (z. B. anderes Gebäude), ist er entweder ohne Verbindung, oder er verbindet sich mit einem anderen verfügbaren WLAN-Sender. Hier muss also die Reichweite eines solchen Netzwerkzugangs berücksichtigt werden (siehe Tabelle 2).

[180] In Anlehnung an Mutschler; Specht 2004, S. 20.

Zelle	Netzwerk / Abdeckung	Reichweite
PAN	Personal Area Network	0-5 m
LAN	Local Area Network	5-1.000 m
MAN	Metropolitan Area Network	1-10 km
WAN	Wide Area Network	10-10.000 km
SAN	Satellite	Global (theoretisch)

Tabelle 2: Verschiedene Zellen und ihre Ausdehnung[181]

Bei einem Mobiltelefon ist das in der Regel anders. Man kann ein Mobiltelefon fast überall hin mitnehmen und ist doch ständig mit dem Netzwerk verbunden. Selbst Ländergrenzen sind durch Roaming (dt. für „Wandern“) heute kein Hindernis mehr. Daher kann man sagen, dass ein Mobiltelefon wirklich mobil ist. Selbst wenn man alle Gebäude einer Stadt mit WLAN ausstatten würde, könnte man bei einer entsprechenden Regelung für den Zugang zwar den besagten Computer überall in der Stadt bewegen, doch abgesehen von einer Verschiebung der Zellart (LAN → MAN) ändert es nichts an der Tatsache, dass man sich nur in diesem physisch begrenzten Netz bewegen kann. Analog gilt das auch für andere Netzwerktechnologien, wie z. B. Bluetooth, Infrarot etc..

Im Umfeld des Mobile Computing unterscheidet Roth[182], aufbauend auf der Arbeit von Pandya[183], drei Arten von Mobilität: Endgerätemobilität, Benutzermobilität und Dienstmobilität (siehe Abbildung 17). Hinzu kommt noch der wichtige Aspekt der Mobilität von Informationen und Wissen, unabhängig von bereitgestellten Diensten. Es können Informationen lokal auf mobilen Endgeräten gespeichert werden oder per Kommunikationsnetze als Dienste auf dem Endgerät verfügbar gemacht werden. Diese Unterscheidung ist für die Realisierung mobiler Anwendungssysteme wichtig, um z. B. zwischen Online- und Offlinedatenhaltung bzw. Online- und Offlineprogrammhaltung zu unterscheiden[184].

[181] In Anlehnung an Tanenbaum 1998, S. 25 ergänzt um PAN und SAN.
[182] Vgl. Roth 2002, S. 7.
[183] Vgl. Pandya 2000.
[184] Vgl. Mutschler; Specht 2004, S. 4.

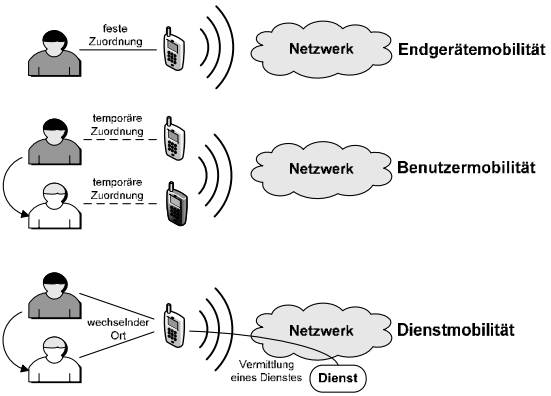

Abbildung 17: Mobilitätsarten[185]

Endgerätemobilität

Endgerätemobilität impliziert ein Gebrauchen des Endgerätes im nicht stationären Fall. Zentral sticht das Merkmal der ortsunabhängigen Benutzung hervor. Eine einfache und mühelose Herstellung und Aufrechterhaltung der Endgerätemobilität für eine angemessene Zeit sind damit Anforderungen an ein „mobiles Endgerät". Somit ist Endgerätemobilität bei einem PC nicht gegeben. Obwohl ein solcher Rechner durchaus „mobil" ist – man kann ihn tragen –, wird er doch während des Herumtragens nicht funktionieren. Dazu müsste er bspw. mit Akkumulatoren, Bildschirm und kabellosem Netzwerkzugang ausgestattet sein, aber dann würde es sich wohl eher um einen speziellen tragbaren Computer (Notebook) handeln. Bei Endgerätemobilität wird üblicherweise eine feste Zuordnung des Benutzers zu einem mobilen Endgerät vorausgesetzt. Über dynamische Verbindungen ist das mobile Endgerät in der Lage, sich mit einem Netzwerk selbsttätig zu verbinden. Hierzu sind in der Regel drahtlose Netzwerke erforderlich.

Benutzermobilität

Bei der *Benutzermobilität* hat der Benutzer die Möglichkeit, unterschiedliche Zugangsgeräte zu nutzen. Ein sich bewegender Benutzer wird nicht vorausgesetzt. Der Benutzer kann sich stationär aufhalten, ist aber nicht an einen besonders ausgezeichneten Ort bzw. an ein bestimmtes Endgerät gebunden. Hierfür muss ein Benutzer über ein unverwechselbares Merkmal zur Identifikation gegenüber dem Netzwerk verfü-

[185] In Anlehnung an Roth 2005, S. 8.

gen[186]. In diesem Sinne ist ein mobiler Nutzer auf das Vorhandensein entsprechender Infrastrukturen an den jeweiligen Nutzungsstätten angewiesen[187].

Dienstmobilität

Auch Dienste können als „mobil" aufgefasst werden, wenn sie sich von Endgerät zu Endgerät übernehmen lassen, oder wenn sie von überall aus erreicht werden können im Sinne der Omnipräsenz. Bei *Dienstmobilität* kann ein Benutzer immer auf dieselben Dienste zugreifen, unabhängig von welchem Ort aus ein Dienst genutzt wird. Somit stehen die Dienste selbst im Mittelpunkt, die nicht mobil sind, jedoch einen mobilen Zugriff unterstützen[188]. Ein Beispiel hierfür ist der weltweite Zugriff auf die eigenen E-Mails. Hierbei ist die eindeutige Adressierbarkeit des Dienstes eine wesentliche Voraussetzung für Dienstmobilität.

3.1.3.3 Mobile Computing

Rund um den Begriff „Mobile Computing" entstehen (neue) technische und fachliche Konzepte der mobilen Anwendungsdomäne. Ein Schwerpunkt ist die Möglichkeit Geschäftsprozesse mobilfähig zu machen[189]. Andererseits lassen sich gerade für den privaten Bereich Anwendungsfelder durch „Mobile Computing" erschließen[190]. Meist bezeichnet „Mobile Computing" die technologische Seite mobiler Anwendungssysteme, also die mobile Technologie[191].

Zimmerman definiert Mobile Computing wie folgt: *„The term ‚Mobile Computing' is used to describe the use of computing devices – which usually interact in some fashion with a central information system – while away from the normal, fixed workplace. Mobile computing technology enables the mobile worker to: (a) create; (b) access; (c) process; (d) store; and (e) communicate information without being constrained to a single location.*[192]*"*. Hierbei geht es ganz allgemein um die Unterstützung der Anwender durch Computer ohne Bindung an bestimmte Orte.

Im Umfeld des Mobile Computing werden verschiedene Begriffe verwendet, deren Abgrenzung oder scheinbare Synonymität Verwirrung schafft. Besonders häufig fallen

---

[186] Vgl. Roth 2005, S. 7.
[187] Vgl. Mutschler; Specht 2004, S. 21.
[188] Vgl. Roth 2005, S. 7.
[189] Vgl. Mutschler; Specht 2004, S. 4.
[190] Vgl. Lonthoff; Ortner 2006a, S. 187.
[191] Vgl. Hanhart; Legner; Österle 2005, S. 46.
[192] Zimmerman 1999, S. 3.

neben Mobile Computing die Begriffe *Ubiquitous Computing, Pervasive Computing* und *Nomadic Computing*[193]. Diese vielfach willkürlich gebrauchten Begriffe werden im Folgenden näher erläutert und voneinander abgegrenzt (siehe Abbildung 18).

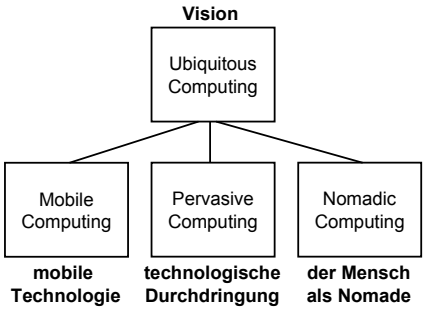

Abbildung 18: Begriffsabgrenzungen im Mobile Computing

Ubiquitous Computing

Der Pionier auf dem Gebiet des Ubiquitous Computing (lat. ubique = überall, Abk.: UC) war Weiser mit seinem Artikel aus dem Jahre 1991 „The computer of the 21th century". Ein kurzer Auszug gibt seine Vision von „allgegenwärtigen Computern" wieder: *„Therefore we are trying to conceive a new way of thinking about computers in the world, one that allows the computers themselves to vanish into the background. […] 'Ubiquitous Computing' in this context does not just mean computers that can be carried to the beach, jungle or airport.*[194]*"*

Weiser beschreibt UC als dritte Phase der Computernutzung nach den Phasen Mainframe und Personal Computer (siehe Abschnitt 1.1). In dieser Phase soll der Computer als Gerät in den Hintergrund treten, ja sogar ganz aus der Wahrnehmung des Menschen verschwinden. Für jede Person werden in der Umgebung viele Computer eingebettet, die vernetzt arbeiten und ein „Netz der Dinge" bilden[195]. Weiser spricht in diesem Zusammenhang auch von „Calm Computing", „Invisible Computing", Disappearing Computing" und meint damit, dass im 21. Jahrhundert die technologische Revolution das Alltägliche, Kleine und Unsichtbare sein wird[196]. Trends in diese Richtung sind z. B. der „intelligente Kühlschrank" oder „intelligente Kleidung", Bordcomputer

[193] Vgl. Mutschler; Specht 2004, S. 18-19.
[194] Weiser 1991.
[195] Vgl. Mattern; Fleisch 2005.
[196] Vgl. Weiser 1991.

im Auto oder auch „intelligente Anwendungen“, die aus dem Verhalten des Benutzers lernen und sich diesem anpassen. Aus technologischer Sicht beschreibt Weiser UC als Framework für neue und interessante Entwicklungen über das gesamte Spektrum der Informatik[197].

Demnach kann Mobile Computing als technologischer Teilbereich von UC gesehen werden. Während MC primär die Mobilität und Transportabilität der Endgeräte zum zentralen Begriff hat, geht UC einen wesentlichen Schritt weiter und setzt das Vorhandensein der Endgeräte an allen Plätzen voraus. Aus diesem Grund kann man UC als die zentrale Vision im Hinblick auf Evolutionsschritte und Weiterentwicklungen im Umfeld mobiler Technologien und Anwendungen bezeichnen.

Mensch : Computer	Beispiel
n:1	Mainframe
1:1	Personalcomputer
1:n	Mobile Endgeräte
n:m	Versteckte Geräte (Ubiquitous/Disappearing Computing)

Tabelle 3: Verhältnis zwischen Mensch und Computer[198]

Ein weiterer wichtiger Unterschied besteht im Verhältnis zwischen der Anzahl der Endgeräte und Anwender (siehe Tabelle 3). Bei klassischen Anwendungen besitzt in der Regel eine Person ein mobiles Endgerät, also im Prinzip eine Relation wie bei dem Personalcomputer. UC hingegen verändert das Verhältnis von 1:1 zu n:m. Das bedeutet, dass viele Menschen sich viele Endgeräte teilen können und sollen. Vor allem gehört im Idealfall keiner Person eines der Endgeräte. Beispiele dafür wären Bildschirme (Anzeigen) in Zügen oder Bussen, Computer im Flur einer Firma usw.

Pervasive Computing

„Pervasive Computing“ (lat. pervadere = durchdringen) bezeichnet nach Hansmann et al.[199] die alles durchdringende Vernetzung von „intelligenten Gegenständen“ des Alltags sowie die Durchdringung des Menschen mit Informations- und Kommunikationstechnologie. Als „Pervasive Computing“ werden dabei die konkreten Ausprägungen der durch die Industrie aufgegriffenen Idee des UC bezeichnet, wobei bereits verfügbare Technik eingesetzt wird, um Geschäftsprozesse und allgemeine Lebensbereiche

[197] Vgl. Weiser 1993.
[198] In Anlehnung an Roth 2005, S. 4f.
[199] Vgl. Hansmann et al. 2001, S. 16.

zu durchdringen[200]. Der Begriff „Pervasive Computing" wird daher häufig fälschlicherweise synonym mit Ubiquitous Computing gebraucht. In einer Studie über Auswirkungen von Pervasive Computing des Zentrums für Technologieabschätzung in Bern[201] wird „Pervasive Computing" als heutige Realisierungen der Visionen des UC möglichst umfassend verstanden, insbesondere sind demnach verwandte Konzepte wie „Ambient Intelligence", „Wearable Computers", „The Invisible Computer" und „Anytime, Anywhere Computing" damit abgedeckt. Als Beispiele werden häufig Wearables (Computer in Kleidung eingebettet), Head-up-Displays und Waschmaschinen mit Controllern genannt. Pervase Computing ist somit eine Vorstufe zur Realisierung einzelner Visionen des UC unter Verwendung heute verfügbarer Technologie.

Nomadic Computing

Aufbauend auf den Grundideen von UC mit Hilfe der Technologien von MC und in Anbetracht der Tatsache, dass Menschen in allen Lebensbereichen mobiler werden (müssen), rückt bei Nomadic Computing (NC) der Mensch in den Mittelpunkt der Betrachtungen. Treffend formuliert es Kleinrock: *„We are all nomads, but we lack the systems support to assist our various forms of mobility. […] As nomads, we own computers and communication devices that we carry about with us in our travels. Moreover, even without carrying portable computers or communications, there are many of us who travel to numerous locations in our business and personal lives, and who require access to computers and communications when we arrive at our destinations.* [202] *„*

3.1.3.4 Merkmale mobiler verteilter Systeme

Ein verteiltes System ist allgemein gesprochen eine Gruppe mehrerer unterschiedlicher Computer (sogenanter Hosts), die miteinander über ein Netzwerk verbunden sind. Auf den Hosts liegen Anwendungen, die miteinander interagieren um Daten auszutauschen oder Dienste anzubieten. Die Interaktion wird dabei von einer Schicht (Middleware) unterstützt, die zwischen verteilten Anwendungen und Betriebssystemkomponenten angeordnet ist.

---

[200] Vgl. Mattern; Fleisch 2005, S. 40.
[201] Vgl. Hilty et al. 2003, S. 27.
[202] Kleinrock 1996, S. 351.

Nach Capra et al.[203] kann die Einteilung verteilter Systeme anhand von drei wesentlichen Kriterien erfolgen. Diese drei Kriterien sind Geräteart, Art der Netzwerkverbindung und Art des Ausführungskontextes.

Art des Gerätes

- *Stationäre Endgeräte*
  Die stationären Endgeräte sind in der Regel leistungsfähig, d. h. sie verfügen über einen großen Speicher und einen schnellen Prozessor.

- *Mobile Endgeräte*
  Die mobilen Endgeräte weisen in der Regel eine geringere Prozessorleistung, weniger Speicher, begrenzte Energiespeicherkapazität und eine kleinere Bildschirmfläche auf.

Art der Netzwerkverbindung

- *Dauerhafte Verbindung*
  Endgeräte können dauerhaft an ein Netzwerk durch ununterbrochene Verbindungen von konstanter Bandbreite angeschlossen werden. Trennungen vom Netzwerk werden als Ausnahmen des normalen Systemverhaltens behandelt. Klassische stationäre verteilte Systeme werden per Kabel dauerhaft mit einem Netzwerk verbunden.

- *Zeitweilige Verbindung*
  Endgeräte können über drahtlose Verbindungen an ein Netzwerk angeschlossen werden. Solche Verbindungen sind nicht immer oder überall vorhanden und die Bandbreiten können jederzeit schwanken. So können z. B. WLAN-Netzwerke angemessene Bandbreiten erzielen, doch nur unter der Bedingung, dass sich die Endgeräte in einer Reichweite von einigen 100 m von ihrer Basisstation befinden. Weiterhin kann die Bandbreite plötzlich drastisch bis auf Null abfallen, so dass die Verbindung mit dem Netzwerk verloren gehen kann, wenn die Endgeräte in einen Bereich ohne Abdeckung oder mit vielen Störungen bewegt werden. Unvorhersehbare Trennungen können nicht mehr als Ausnahme betrachtet werden, sondern sie werden Teil der mobilen Kommunikation. Mobile verteilte Systeme sind meist zeitweilig mit drahtlosen Netzwerken verbunden.

---

[203] Vgl. Capra; Emmerich; Mascolo 2002, S. 23.

Art des Ausführungskontextes

In diesem Zusammenhang beschreibt der Begriff Kontext die Gesamtheit aller Einflüsse, die sich auf das Verhalten und die Ausführung einer Anwendung auswirken können. Diese Definition schließt interne Betriebsmittel, wie Speicher oder Bildschirmfläche und externe Betriebsmittel, wie Bandbreite, Qualität der Netzwerkverbindung, Position, benachbarte Endgeräte oder Dienste ein.

- *Statischer Ausführungskontext*
  In einer stationären verteilten Umgebung ist der Ausführungskontext grundsätzlich statisch. Die Bandbreite ist hoch und die Verbindung kontinuierlich. Die physische Position der Endgeräte wird äußerst selten geändert, da sie meist groß, schwer bzw. teuer sind. Weiterhin kann das Auffinden von Diensten (engl. Service Lookup), die im Netzwerk angeboten werden, erleichtert werden, indem der Dienstanbieter sich an einen zentralen Registrierungsdienst anmeldet.

- *Dynamischer Ausführungskontext*
  Der Kontext von mobilen verteilten Systemen ist dynamisch. Endgeräte können schnell in das Netzwerk eintreten und auch schnell aus dem Netzwerk verschwinden, so dass Dienste, die vor der Trennung der Verbindung registriert waren, nach einem Wiederanschluss nicht mehr erreicht werden können. Dienstlokalisierung wird in einer mobilen Umgebung aufgrund der fehlenden festen Infrastruktur der mobilen Geräte erschwert. Feste Annahmen über die Konfiguration des Kontextes können nicht getroffen werden, so dass kein bestimmter Server für die Lokalisierung von Diensten identifiziert oder kontaktiert werden kann. Weiterhin kann die Position der Endgeräte nicht als fest angenommen werden, da mobile Endgeräte schon aufgrund ihrer Nutzung häufig ihre Position wechseln. Weiterhin können Bandbreite und Qualität der Netzwerkverbindung stark schwanken.

Anhand der hier aufgeführten Merkmale lassen sich sämtliche heute existierende Möglichkeiten der Vernetzung mobiler und nicht mobiler Endgeräte allgemein einteilen. Entsprechend diesen Kriterien können drei Kategorien von verteilten Systemen unterschieden werden[204]: traditionelle verteilte Systeme, ad-hoc verteilte Systeme und nomadisch verteilte Systeme.

[204] Vgl. Capra; Emmerich; Mascolo 2002, S. 25.

Traditionelle verteilte Systeme

Das Prinzip der traditionellen verteilten Systeme beruht auf einer Gruppe stationärer Geräte, die dauerhaft an ein zuverlässiges Netzwerk angeschlossen sind (siehe Abbildung 19).

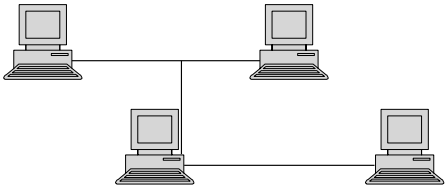

Abbildung 19: Prinzip der traditionellen verteilten Systeme[205]

Sie befinden sich in einem statischen Umfeld und verfügen über hohe Bandbreiten sowie beständige Verbindungen. Infrastrukturen wie z. B. Middleware, die in einem solchen statischen Umfeld entwickelt werden und auf denen verteilte Anwendungen aufgebaut werden, müssen nach Capra et al.[206] folgende Anforderungen erfüllen:

- *Skalierbarkeit*
  Hierbei ist zwischen vertikaler und horizontaler Skalierbarkeit zu unterscheiden. Vertikale Skalierbarkeit bezeichnet das Verhalten von Programmen oder Algorithmen bezüglich des Ressourcenbedarfs bei wachsenden Eingabemengen (Performance und Komplexität). Horizontale Skalierbarkeit hingegen bezeichnet positive Auswirkungen des Gesamtsystems bei Erhöhung des Ressourceneinsatzes.

- *Offenheit*
  Eigenschaft der Anpassungsfähigkeit eines Systems, z. B. durch Erweiterungen oder Änderungen, um veränderten Funktionsanforderungen zu entsprechen.

- *Heterogenitätsüberwindung*
  Hierbei kann zwischen verschiedenen Formen der Heterogenität unterschieden werden (siehe Abschnitt 2.5.1.1). Die Überwindung kann bspw. durch Übersetzen unterschiedlicher Sprachen bzw. Protokolle (Zwischensprachenkonzept) erreicht werden, um Integration verschiedener Komponenten in unterschiedliche Betriebssysteme und Hardwareplattformen zu erreichen.

---

[205] In Anlehnung an Capra; Emmerich; Mascolo 2002, S. 26.
[206] Vgl. Capra; Emmerich; Mascolo 2002, S. 25.

- *Fehlertoleranz*
  Fähigkeit eines Systems, Störungen zu behandeln oder aufzuheben, ohne den Betrieb des ganzen Systems anzuhalten.

- *Benutzung gemeinsamer Ressourcen*
  Unterschiedliche Anwender bzw. Anwendungen eines verteilten Systems können gemeinsam die verschiedenen Hardware- und Softwareressourcen (z. B. Drucker, Datenbank etc.) des Systems nutzen.

Traditionelle verteilte Systeme waren die erste Form der verteilten Systeme. Erfolgreiche Middleware-Technologien, welche die Kommunikation zwischen unterschiedlichen Anwendungen und damit eine Anwendungsintegration erlauben, sind bereits erfolgreich hierfür entworfen und eingesetzt worden[207].

Ad-hoc mobile verteilte Systeme

Ad-hoc mobile verteilte Systeme bestehen aus einer Menge von mobilen Geräten, die durch eine zeitweilige Verbindung von variabler Qualität an das Netz angeschlossen sind und in einem extrem dynamischen Umfeld ohne feste Infrastruktur arbeiten[208]. Sie unterscheiden sich von den traditionellen verteilten Systemen dadurch, dass sich mobile Geräte vollständig isolieren und sich unabhängig von den anderen entwickeln können. Mobile Netzwerktechnologien, wie bspw. Bluetooth, erleichtern die komplizierten Konfigurationen mit mehrfachen sogenannten Piconets[209], deren Integration Scatternets[210] bilden[211]. Ad-hoc Netze stellen die höchsten Anforderungen an Flexibilität im Umfeld des Mobile Computing dar.

Anforderungen, die ursprünglich für traditionelle verteilte Systeme formuliert wurden, bleiben auch im Fall der ad-hoc Netzwerke bestehen. Skalierbarkeit bspw. bleibt weiterhin eine wichtige Anforderung, da eine gegebene Netzwerkbandbreite unter den Geräten innerhalb des gleichen Netzwerks bzw. der gleichen „Wolke“ (siehe Abbildung 20) geteilt werden muss.

---

[207] Vgl. Capra; Emmerich; Mascolo 2002, S. 25.
[208] Vgl. Kehr 2000.
[209] Piconet bezeichnet ein kleines Netzwerk aus Endgeräten ohne zentrale Verwaltung.
[210] Mit Scatternet wird eine Gruppe überlappender unabhängiger Piconets bezeichnet, die mindestens einen gemeinsamen Netzwerkteilnehmer enthalten.
[211] Vgl. Bluetooth SIG 2003

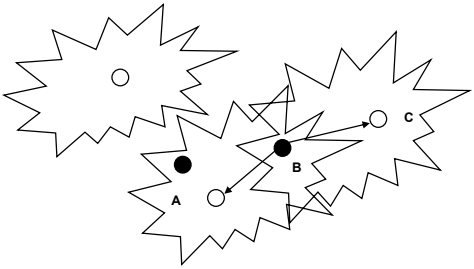

Abbildung 20: Prinzip der ad-hoc mobilen verteilten Systeme[212]

Zusätzlich zu den bereits genannten Anforderungen, kommen bei den ad-hoc verteilten Systemen neue, besondere Herausforderungen hinzu, wie z. B. die Heterogenitätsüberwindung. Während stationäre Geräte in einem festen Netzwerk eingebunden bleiben, treffen mobile Geräte auf mehrere heterogene Verbindungen und Anwendungen, da sie ständig den Abdeckungsbereich eines Netzwerks verlassen und in einen anderen wechseln können. Auch die Vielfalt mobiler Endgeräte steigt durch ständige technologische Weiterentwicklung.

Aufgrund der geänderten physischen Infrastruktur können die Middleware-Lösungen, die für traditionelle verteilte Systeme geeignet sind, nicht ohne weiteres erfolgreich auf die mobile Umgebung übertragen werden.

Nomadisch verteilte Systeme

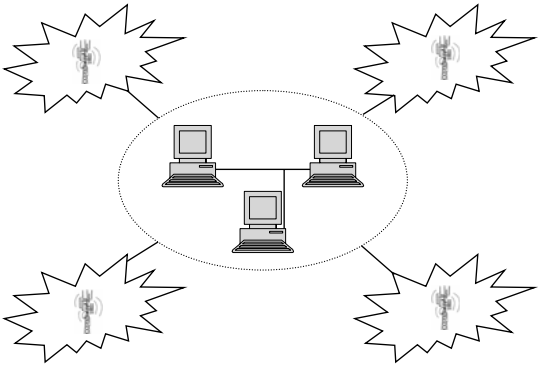

Abbildung 21: Prinzip der nomadischen verteilten Systeme[213]

[212] In Anlehnung an Capra; Emmerich; Mascolo 2002, S. 27.
[213] In Anlehnung an Capra; Emmerich; Mascolo2002, S. 25.

Nomadisch verteilte Systeme sind zwischen traditionellen festen Netzwerken und ad-hoc Netzwerken einzuordnen und verbinden einige Merkmale beider Gruppen[214]. Nomadische Systeme basieren auf einem stabilen Kern (siehe Abbildung 21) von stationären Geräten, wie bspw. Router, Switches etc.. Am Rande des festen Netzwerks befinden sich Basisstationen, die über Möglichkeiten zur drahtlosen Kommunikation verfügen. Sie kontrollieren und leiten Nachrichten von und zu den mobilen Geräten weiter.

3.2 Der Begriff „Architektur“

Durch seinen häufig undifferenzierten Gebrauch, soll der Begriff „Architektur“ im Umfeld der (Wirtschafts-)Informatik zunächst einmal geklärt werden. Aus dem Umfeld der Datenbank-Systeme wird der Architekturbegriff für Ebenen-Architekturen im Sinne von Abstraktionsebenen verwendet. Eine solche Ebeneneinteilung auf Basis von Abstraktion wurde erstmals mit dem Data Independent Access Model[215] (DIAM) vorgeschlagen. Eine frühe und zugleich grundlegende Architektur relationaler Datenbank-Systeme findet sich in der ANSI/SPARC-Architektur[216], auch Drei-Ebenen-Architektur genannt. Diese Architektur wurde 1975 vom Standards Planning and Requirements Committee (SPARC) des American National Standards Institute (ANSI) entwickelt[217]. In dieser Architektur werden externe, konzeptionelle und interne Ebenen unterschieden[218]. Ein wesentliches Ziel hierbei ist die Datenunabhängigkeit. Hierdurch wurde eine Trennung von Daten und Programmen möglich.

Auch der Begriff „Software-Architektur“ ist erklärungsbedürftig. Ausgehend von frühen Arbeiten von Shaw[219], Schwanke[220] und Perry[221] wurde der Architekturbegriff in die Software-Engineering Literatur eingeführt. Eine umfangreiche Übersicht von möglichen Definitionen des Begriffs „Software-Architektur“ findet sich auf einer Webseite des Software Engineering Institute (SEI) der Carnegie Mellon University, Pittsburgh (USA)[222]. Einige ausgewählte Definitionen sollen hier angeführt werden:

---

[214] Vgl. Capra; Emmerich; Mascolo 2002, S. 25.
[215] Vgl. Senko et al. 1972.
[216] Tsichritzis; Lug 1978.
[217] ANSI/X3/SPARC 1975.
[218] Vgl. Härder; Rahm 1999, S. 8ff.
[219] Shaw 1989.
[220] Schwanke; Altucher; Platoff 1989.
[221] Perry; Wolf 1992.
[222] SEI 2006.

„[...] beyond the algorithms and data structures of the computation; designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives.[223]“

„An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization – these elements and their interfaces, their collaborations, and their composition.[224]“

„The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.[225]“

„The architecture of a complex software system is its style and method of design and construction. When applied appropriately, a good software architecture facilitates application system development, promotes achievement of the system's functional requirements, and supports reconfiguration.[226]“

Den meisten hier aufgeführten Definitionen liegen zwei Gemeinsamkeiten zugrunde. Zum einen sind Architekturen Beschreibungen des Aufbaus von Software(systemen) aus Komponenten bzw. Modulen. Zum anderen werden die Relationen zwischen den einzelnen Komponenten bzw. Modulen in einer Architektur beschrieben. Somit beschreibt eine Architektur zumindest die grundlegenden Elemente und deren Struktur.

Eine zweckmäßige Definition findet sich in der Beschreibung des IEEE-Standards 1471-2000 zur Software-Architekturbeschreibung[227]: *„Die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, deren Beziehungen zueinander*

[223] Garlan; Shaw 1993.
[224] Kruchten 2004.
[225] Bass; Clements; Kazman 2003.
[226] Hayes-Roth et al. 1995.
[227] IEEE 2000.

und zur Umgebung sowie den Prinzipien, die den Entwurf und die Evolution des Systems bestimmen.[228] "

Der Architekturbegriff kann weiter differenziert betrachtet werden. Eine Orientierung erfolgt hierbei an dem Architekturbegriff, wie er im Bauwesen verwendet wird. In diesem Bereich haben sich Architekturmuster und -stile (siehe Abschnitt 3.2.1) etabliert sowie eine perspektivische Betrachtung von Architekturen. In Anlehnung an Grundrisse, Querschnittpläne und räumliche Modelle kann die Architektur eines Anwendungssystems unter verschiedenen Blickwinkeln betrachtet werden (siehe Abschnitt 3.2.2). Siedersleben[229] spricht von einer Außensicht des Systems und meint damit die Spezifikation (Geschäftsprozesse, Datenmodell, Funktionsmodell, Dialog-Benutzungsschnittstelle, Batch-Benutzungsschnittstelle und Nachbarsysteme) und einer Innensicht als das Ergebnis der Konstruktion (Komponentensicht, Betreibersicht, Erstellungssicht, Physische Schicht und Laufzeitschicht).

Diese differenzierte Betrachtung von Architektur führt zu dem Prinzip der Abstraktion[230], im Sinne der Gleichheit zwischen verschiedenen Dingen in gewisser Hinsicht. Hierbei kann eine Gleichheit (Invarianz) bezüglich des fachlichen, logischen und physischen Aufbaus eines Softwaresystems herausgestellt werden. Die Abstraktion wird zur Schaffung abstrakter Objekte und damit als Mittel zur Komplexitätsreduktion eines Gesamtsystems eingesetzt.

Die Abstraktionskriterien für Anwendungsarchitekturen unterscheiden sich von denen für Basissystemarchitekturen (siehe Abschnitt 3.2.3 und Abschnitt 3.2.4). Dabei ist die Wahl einer geeigneten Granularität der einzelnen zu betrachtenden Komponenten ein Hauptproblem des Architekten. Ein Kriterium für geeignete Modul-/Komponentenbildung ist die funktionale Disjunktheit, d. h. eine Funktion ist in genau einer Komponente realisiert. Eine Funktion ist hierbei ein Abstraktor[231], d. h. mit einer Funktion wird ein abstrakter Gegenstand bezeichnet.

---

[228] Hasselbring 2006.
[229] Siedersleben 2003, S. 89-91.
[230] Vgl. Wedekind; Ortner; Inhetveen 2004.
[231] Vgl. Wedekind; Ortner; Inhetveen 2004, S. 338f.

3.2.1 Architekturmuster und -stile

Als „Mikro-Architekturen“ können Entwurfsmuster gezählt werden. Im Bauwesen gibt es hierfür ein Standardwerk von Alexander[232], in dem genau 251 verschiedene Entwurfsmuster für die Gestaltung von Städten, Gebäuden und Bauwerken beschrieben und empfohlen werden. Aus den Beiträgen von Alexander[233,234] wurden Konzepte für das entwurfsmusterbasierte Software Engineering[235] kombiniert und adaptiert. Ein Beispiel für ein Entwurfsmuster aus dem Gebiet der Anwendungssysteme ist das Composite-Muster für den objektorientierten Entwurf. Es beschreibt eine bewährte Struktur (Abstraktor) für die hierarchische, rekursive Strukturierung zusammengesetzter Objekte[236].

Architekturmuster hingegen beschreiben den Stil der Gesamtarchitektur eines Systems. Bewährte Architekturmuster im Bereich von Anwendungssystemen sind die hierarchische Schichtenarchitektur (auch Sprachschichtungen), die Client/Server-Architektur, die Peer-to-Peer-Architektur, die Service-orientierte Architektur und Komponentenarchitektur[237].

3.2.2 Mittel zur Architekturbildung

Vom Standpunkt des Software-Engineering stehen zur Architekturbildung Komposition, Klassifikation, Modularisierung, Aspektierung und Ebenenbildung als Mittel zur Verfügung, die im Folgenden kurz beschrieben werden.

Komposition

Die Komposition basiert auf der Abhängigkeit von Gegenständen. *„Die Komposition fasst Gegenstände zu neuen Gegenständen zusammen, um die Eigenschaften, welche sich aus der Abhängigkeit oder Verbindung dieser (Teil-)Gegenstände ergeben, in einem Ganzen zu beschreiben.[238]“* Durch eine solche Komposition gebildete Ganzheiten sind keine abstrakten Gegenstände, wie bspw. Mengen oder Klassen, sondern sind Gegenstände der gleichen Sprachebene wie ihre Teile.

---

[232] Alexander et al. 1977.
[233] Alexander 1979.
[234] Alexander 1999.
[235] Vgl. Gamma 2005.
[236] Vgl. Hasselbring 2006, S. 48.
[237] Vgl. Hasselbring 2006, S. 48-49.
[238] Schienmann 1997, S. 59.

Komposition eignet sich, um Zusammenhänge, wie Teil-Ganze-Beziehungen (mereologische Zusammenhänge[239]), existenzielle, funktionale, einseitige und wechselseitige Abhängigkeiten auszudrücken[240].

Klassifikation

Die Klassifikation basiert auf der Gleichheit von Gegenständen. Klassifikation ist ein Verfahren zur Einordnung verschiedener Begriffswörter gemäß bestimmter Kriterien unter einem (neuen) Begriffswort.

Klassifikationen stellen systematische, meist partielle Ordnungen (bspw. Art-Gattungs-Beziehungen[241]) von Bereichen dar, welche nach bestimmten Ordnungsprinzipien hergestellt werden. Der untersuchte Wirklichkeitsausschnitt („Realität“) sollte von einer Klassifikation „wirklichkeitsgetreu“ widergespiegelt werden. Friedell merkt hierzu kritisch an, dass alle Klassifikationen, die der Mensch erstellt hat willkürlich, künstlich und falsch, aber trotzdem nützlich, unentbehrlich und unvermeidlich sind, weil Menschen in Ordnungen denken[242]. Grundsätzlich dienen Klassifikationen dazu, einen betrachteten Wirklichkeitsausschnitt möglichst schnell zu konstruieren und überblicken zu können und es sollen mit den enthaltenen Beziehungen Orientierungspfade vorgefunden werden. Damit verbunden ist die Notwendigkeit, insbesondere in Bereichen, die einer andauernden Veränderung unterliegen, Klassifikationen im Laufe der Zeit anzupassen, weiterzuentwickeln oder sogar durch eine vollständig neue Klassifikation zu ersetzen. Neben einer grundsätzlichen Unterscheidung zwischen analytischer Klassifikation (vom Allgemeinen zum Besonderen – deduktive Vorgehensweise) und synthetischer Klassifikation (vom Besonderen zum Allgemeinen – induktive Vorgehensweise) lassen sich Klassifikationen an Hand grundsätzlicher Leistungsmerkmale sowie nach identifizierbaren Vor- und Nachteilen differenzieren.

Modularisierung

Die Grundlage modularer Systeme geht auf das von Parnas entwickelte Geheimnisprinzip (engl. Information hiding) zurück[243]. Die Modularisierung ist ein Verfahren für die Zerlegung eines Software-Systems in einzelne Module. Eine Entscheidung, welche Teile in einem Modul zusammengefasst werden und welche Teile in unterschiedliche

[239] Vgl. Ridder 2002.
[240] Vgl. Schienmann 1997, S: 60.
[241] Vgl. Heinemann 2006, S. 109.
[242] Vgl. Friedell 1960, S. 5.
[243] Vgl. Parnas 1972, 1056.

Module aufgeteilt werden, soll unter Berücksichtigung der Aspekte Geschlossenheit, Bindung, Minimalität der Schnittstelle, Testbarkeit, Inferenzfreiheit, Import- und Verwendungszahl erfolgen[244].

(orthogonale) Aspektierung

Aspektierung bezeichnet ein Verfahren, bei dem Teile und deren Zusammenhänge unter verschiedenen Aspekten getrennt betrachtet werden können[245]. Ein Architekt spezifiziert ein Haus nicht vollständig in einem einzigen Plan, sondern er fertigt Struktur- und Detailpläne an. In analoger Weise erfolgt eine solche aspekteorientierte Beschreibung im Software-Engineering. Hierdurch wird vermieden, dass alle Details in nur einem Plan beschrieben werden müssen. Bestimmte Bestandteile werden getrennt von anderen Aspekten spezifiziert. Dabei können die Aspekte unabhängig (orthogonal) voneinander sein. In der Aspektorientierten Programmierung[246] (AOP) wird diese Aspektierung zum Programmierparadigma erhoben.

Ebenenbildung

Unter Ebenenbildung (allgemein Stratifikation[247]) wird eine auf Basis verschiedener Prinzipien basierte Einteilung eines Gegenstandsbereichs in Form einer Schichtung erreicht. Als Prinzipien können im Software-Engineering Abstraktion und Sprachschichtung eingesetzt werden.

Abstraktionsebenen

Abstraktion fasst alle Konstruktionshandlungen zusammen, die auf einer Äquivalenzrelation basieren[248]. Die Äquivalenzrelationen können hierbei auf Grundlage der Identität oder der Parität aufbauen. Die Identität verlangt die vollkommene Übereinstimmung der Gegenstände bei gleichzeitiger Verschiedenheit der Begriffe (Abendstern und Morgenstern sind identisch). Parität bezeichnet zum einen Gleichheit von Gegenständen die unter denselben Begriff fallen (Subsumtion, wie bspw. Jörg und Silke im Hinblick auf den Begriff „Mensch“) und zum anderen Begriffe, die untergeordnete Begriffe eines Oberbegriffs sind (Subordination, wie bspw. Notebook und PDA im Hinblick auf den Begriff „Computer“). Gegenstände mit gemeinsamen relevanten Eigenschaften werden einer bestimmten Ebene zugeordnet.

---

[244] Vgl. Siedersleben 2003, S. 97ff.
[245] Vgl. Aßmann; Neumann 2003, S. 22f.
[246] Vgl. Böhm 2006, S. 269.
[247] Vgl. Ortner 2004, S. 149.
[248] Vgl. Ortner 1983, S. 80.

Sprachebenen

Die sprachbasierte Informatik liefert die Konzepte der Sprachebenen und Sprachräume (siehe Abbildung 22) zur wiederspruchsfreien Rekonstruktion von (Anwendungs-)Architekturen.

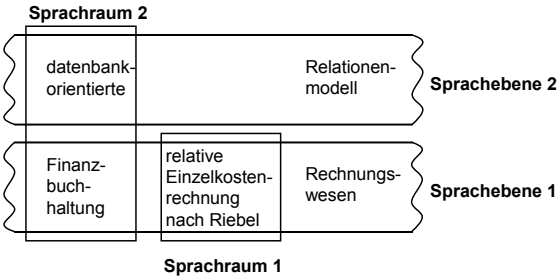

Abbildung 22: Sprachebenen und Sprachräume[249]

Eine Sprachebene umfasst in der Regel zwei Sprachschichten, eine Schema-Schicht und eine Ausprägungs-Schicht (siehe Abschnitt 2.5.1.1), die über Sprachhandlungen verbunden sind. Sprachschichtungen können in einer Sprache oder in mehreren Sprachen (z. B. Objektsprache, Metasprache) zur Erzeugung von Architekturen eingeführt werden. Sprachräume, die über eine Sprachebene oder über mehrere Sprachebenen definiert sein können, sind dagegen thematische (inhaltliche) Einteilungen bestimmter Anwendungsgebiete[250].

Bei der Sprachebenenbildung können durch Betrachtung unterschiedlicher Gegenstandsbereiche verschiedene Varianten auftreten. Hierbei werden die Ebenen Objektebene, Metaebene und Paraebene unterschieden.

Mit Objektsprache wird die Sprachebene bezeichnet auf der man über die Objekte spricht. Die Metasprache ist allgemein eine Sprache über eine Sprache. Im Bezug auf die Objektsprache wird in der Metasprache die Objektsprache beschrieben. Diese Unterscheidung von Sprachstufen wurde von Tarski entwickelt[251]. Die Parasprache, als eine Sprache von nichtsprachlichen Mitteln, liegt als eine Hilfsprache bereits rekon-

[249] Vgl. Ortner 2005a, S. 80.
[250] Vgl. Ortner 2005a, S. 81.
[251] Tarski 1979.

struiert vor[252]. Zwischen diesen Sprachebenen können Referenzbeziehungen hergestellt werden.

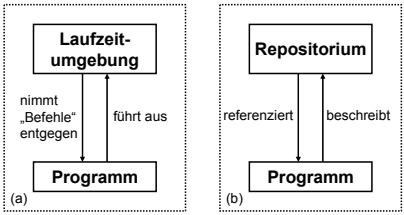

Abbildung 23: Varianten der Sprachebenen

Im Fall einer Laufzeitumgebung, in der ein Programm ausgeführt wird, ist das Programm selbst auf Objektsprachebene und die Laufzeitumgebung auf Parasprachebene einzuordnen (siehe Abbildung 23 (a)). Das Programm nimmt aus der Laufzeitumgebung „Befehle“ entgegen. Die Laufzeitumgebung führt das Programm aus.

In Repositorien werden Gegenstände, wie bspw. Programme, auf der Objektsprachebene und das Repositorium selbst auf Metasprachebene eingeordnet (siehe Abbildung 23 (b)). Im Repositorium werden die Programme beschrieben. Das Programm wird aus dem Repositorium referenziert.

3.2.3 Anwendungsarchitektur

Eine Anwendungsarchitektur beschreibt den fachlichen Aufbau von Komponenten einer Anwendung und die Beziehungen zwischen ihnen. Hierbei erfolgt ausschließlich eine fachliche Betrachtung der Anwendungsbereiche (Domäne), wobei häufig ein Basissystemparadigma, wie bspw. das Datenbank-Management-Paradigma oder Workflow-Management-Paradigma hinzugenommen wird.

Ein Beispiel aus der Domäne Rechnungswesen für betriebliche Anwendungen soll einen übersichtlichen Einstieg in diese Thematik ermöglichen. In dem Beitrag von Ortner[253] wird zur Modellierung eines Datenbankentwurfs für ein Buchungs- und Abrechnungssystem zunächst der fachliche Aufbau – orientiert an der Drei-Schema-Architektur für Daten[254] – des gewünschten Systems in Form einer Anwendungsarchitektur beschrieben.

---

[252] Vgl. Ortner 2005a, S. 248.
[253] Ortner 1984b.
[254] Vgl. Ortner; Söllner 1987, S. 136.

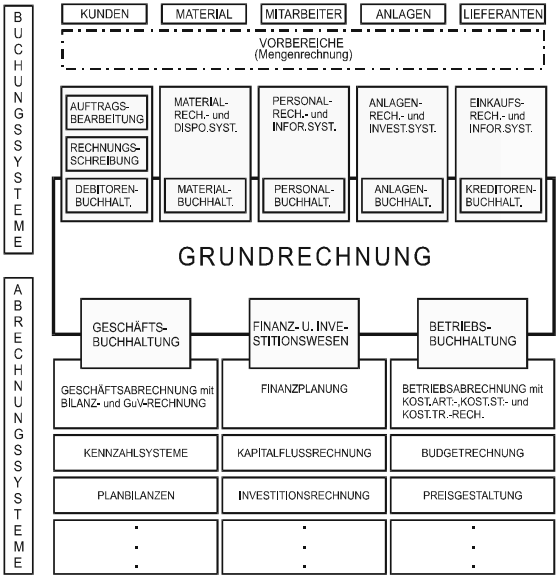

Abbildung 24: Datenbankorientierte Anwendungsarchitektur für das Rechnungswesen[255]

Im Rechnungswesen wird einerseits unterschieden zwischen Nebenbuchhaltungen, wie Material-, Personal- oder Debitorenbuchhaltung und der sie integrierenden Hauptbuchhaltung (Sachkontenbuchhaltung) sowie andererseits zwischen den beiden Auswertungsrichtungen eines solchen Buchungs- und Abrechnungsinformationssystems als pagatorische Rechnung in der Finanz- und Geschäftsbuchhaltung und als kalkulatorische Rechnung in der Kosten- und Leistungsrechnung eines Unternehmens. Daran schließen sich Sonderrechnungen wie Kennzahlenanalyse, Planbilanzen, Budgetrechnung, Preisbildung etc. an (siehe Abbildung 24).

3.2.4 Basissystemarchitektur

Um Missverständnissen bei dem Gebrauch des Begriffs „Basissystemarchitektur“ vorzubeugen, muss hiervon zunächst einmal der Begriff Basisarchitektur abgegrenzt werden. Basisarchitekturen beschreiben die grundlegende Architektur eines Anwendungssystems im Ganzen. Damit ist dieser Begriff mit dem Architekturmusterbegriff verwandt bzw. für den ganzheitlichen Ansatz müsste man von Anwendungssystemarchitektur (siehe Abschnitt 3.2.5) sprechen. Basissystemarchitekturen sind grundlegende Beschreibungen des Aufbaus der Basissysteme. In einem Anwendungssystem kann

[255] In Anlehnung an Ortner 1984b, S. 177.

dabei nicht nur ein einziges Basissystem zum Einsatz kommen, sondern es können auch mehrere nebeneinander (komponentenorientiert) oder übereinander (schichtenorientiert) angeordnete Basissysteme zum Einsatz kommen.

Auch hierbei gibt es grundlegende Architekturstile, wie Schichtung, Client/Server-Architektur bzw. Komponentenarchitektur. Die Wahl der Basissystemarchitektur richtet sich nach den Anforderungen der Anwendungsarchitektur (Voruntersuchung, Pflichtenheft). Auch hierbei existiert eine Schichtung der Architekturen, wie bspw. Anwendungsarchitektur, Basissystemarchitektur, Rechnerarchitektur, Netzwerkarchitektur etc. (siehe Abbildung 25).

Basissysteme bieten den übergeordneten Anwendungen generische Funktionen an. Beispiele für Basissysteme sind: Datenbank-Managementsysteme (DBMS), Workflow-Management-Systeme (WfMS), Expertensysteme etc..

3.2.5 Anwendungssystemarchitektur

Eine Anwendungssystemarchitektur beschreibt den Aufbau und die Zusammenhänge eines gesamten Anwendungssystems, das aus mehreren Schichten aufgebaut ist. In einer groben Übersicht können vier Anwendungssystemarchitekturansätze in ihrer zeitlichen Entwicklung voneinander abgegrenzt werden, wobei die vierte Architektur eine zukunftsweisende Entwicklung ist, die den Begriff „Anwendungssystem“ in seiner weitesten Fassung voraussetzt.

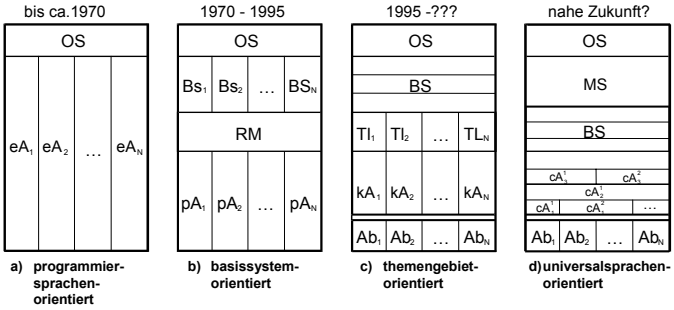

Abbildung 25: Entwicklung von Anwendungsarchitekturen[256]

[256] In Anlehnung an Ortner 2006, Folie 335.

Programmiersprachenorientierung

In dieser frühen Form einer Anwendungsarchitektur bilden die Betriebssysteme der Großrechner (Mainframe) die Basis für einzelne Anwendungen, die funktionsorientiert isoliert auf die Betriebssystemebene zugreifen. Zeitlich fällt diese Architektur in die Zeit der elektronischen Massendatenverarbeitung. Diese Tätigkeit wurde überwiegend durch Experten ausgeführt, die in der Lage waren, die Großrechner zu bedienen.

Basissystemorientierung

Die große Bedeutung bestimmter Funktionen des Betriebssystems (z. B. Prozessverwaltung, Scheduler, Netzwerkverwaltung etc.) für einen integrierten Betrieb der Anwendung hat zur Entwicklung von Basissystemen geführt. Basissysteme realisieren sogenannte „generische Funktionen" (z. B. Verwalten, Steuern, Schlussfolgern etc.). Referenzmodelle (Branchendatenmodelle, Geschäftsprozessmodelle etc.) unterstützen die inhaltliche Integration der Anwendungen und sorgen dafür, dass ausgetauschte Daten auf unterschiedlichen Ausprägungen dieser Architektur gültig und interpretierbar bleiben. Auf Basis der Referenzmodelle sind Paketanwendungen (z. B. für den Einkauf, das Rechnungswesen oder für den Betrieb) dadurch charakterisiert, dass sie mit demselben Basissystem (z. B. einem DBMS) in einem Unternehmen isoliert nebeneinander betrieben werden. Dabei können auf einem Rechner ein oder mehrere Basissysteme installiert sein.

Themengebietorientierung

Durch den Wandel bzw. die Entwicklung der Hardware, die nahezu jeden Gegenstand der physischen Welt an die Informations- und Kommunikationswelt anbindet, ist ein weiterer Wandel der Anwendungsarchitektur notwendig. Zur ganzheitlichen Entwicklung der Anwendungssysteme gehören heute die Rekonstruktion des Wissens der Anwender sowie die Gestaltung der Aufbau- und Ablauforganisation in den Anwendungsbereichen (Organisationsmodellierung). Themengebiete stehen komplementär zum Geschehen in der physischen Welt (Anwendungsbereiche). Eine solche komplexe Architektur der Anwendungen kann komponentenorientiert beherrschbar und administrierbar werden. Für die materiale (inhaltliche) Integration der Anwendungen sind (Fach-)Terminologien flexibler und mit geringerem Aufwand administrierbar als Referenzmodelle. Basissysteme sind schichtenartig aufgebaut. Komponenten-Anwendungen können über verschiedene Terminologien integriert werden. Anwendungsbereiche greifen auf vorhandene Komponenten-Anwendungen zu.

Universalsprachenorientierung

Eine klar nach den Grundlagen der Sprachschichtung aufgebaute Architektur für z. B. „Cyberspace-Anwendungen“ (Internet-Anwendungen) ist die Electronic New Organon {Service, Servant, Server} (E-NOgS³)-Architektur (siehe Abschnitt 3.6.1). Ein Metainformationssystem bildet die Basis für die geschichteten Basissysteme. Auf Grundlage dieser Basissysteme werden (Cyberspace-)Anwendungen entwickelt. Dabei sind die (Cyberspace-)Anwendungen weiter strukturiert durch Sprachebenen und Sprachräume. Durch die Integration des alle Ebenen umfassenden Sprachansatzes (rekonstruierte rationale Zwischensprachen) und die Einbettung eines Reflexionsmechanismuses (inhaltliche Selbsteinordnung) in zahlreiche Komponenten wird in Zukunft der „Cyberspace“ (Internet) als komplementäres Medium der Anwender (sogenannte Netz-Bürger[257]) mit der physischen Welt verbunden. Hierbei liegt die Vision der Überwindung der „babylonischen Sprachverwirrung“ durch die Implementierung eines Zwischensprachenkonzepts zugrunde.

3.3 Anwendungsarchitekturen in mobilen verteilten Systemen

Eine klare Abgrenzung des Anwendungsbereichs (Domäne) wird aufgrund der Dynamik in Forschung und Entwicklung im Bereich mobiler Anwendungssysteme erschwert. Hinzu kommt die Vermischung der Klassifikationsansätze mit der eingesetzten Technologie, wie bspw. Endgeräte und Kommunikation. Klassifikationsansätze auf Metaebene werden kurz in Abschnitt 3.3.1 besprochen. In der Literatur[258] sind Einteilungen nach Endgeräten, Übertragungstechnologien und z. B. erforderliche Speicherkapazitäten in verschiedenen Anwendungsbereichen häufig zu finden und werden als realisierungsnahe Konzepte bezeichnet. Auch wenn diese Konzepte keine Anwendungsarchitekturen im Sinne von Abschnitt 3.2.3 darstellen, sollen sie aufgrund ihrer praktischen Bedeutung und weiten Verbreitung in Abschnitt 3.3.2 mit aufgeführt werden. Weniger klar sind Zuordnungen und Begriffe in abstrakten Konzepten zur Klassifikation von Anwendungsgebieten im mobilen Bereich. Hierauf wird in Abschnitt 3.3.3 eingegangen. Eine Zusammenfassung der Klassifikationsansätze wird in Abschnitt 3.3.4 gegeben.

[257] Vgl. Ortner 2005a, S. 291.
[258] Vgl. Eller 2005, S. 7-17.

3.3.1 Anwendungsbereich auf Metaebene

Ortner[259] unterscheidet nach dem Zweck der Anwendungsentwicklung für bestimmte Zielgruppen zwischen Wirtschaft, Verwaltung und privatem Bereich. Eine ähnliche Unterscheidung wird in dem Projekt mik21[260] mit Wirtschafts-, Verwaltungs- und Alltagssystemen vorgenommen. Roth[261], Merz[262] und Lehner[263] verwenden den Begriff „privater Bereich“ im Zusammenhang mit der Abgrenzung von Zielgruppen (potenzielle Kunden) für bestimmte Anwendungen. Dieser Differenzierung kommt in den genannten Werken allerdings nur eine untergeordnete Bedeutung zu. Eine weitere Abgrenzung dieser Bereiche erfolgt nicht, es wird lediglich eingeräumt, dass eine Abgrenzung zwischen privatem und öffentlichem Bereich aber auch zwischen Wirtschaft und privatem Bereich nicht eindeutig vorgenommen werden kann bzw. hier die Grenzen zunehmend verschwimmen.

Im Projekt mik21[264] werden – ausgehend von dieser übergeordneten Klassifikation im stationären Bereich – den Systemen mobile Anwendungsformen zur Seite gestellt, wobei diese Vorgehensweise aus dem Projektziel (Untersuchung der Migration von stationären in mobile Anwendungsformen) resultiert. Für die mobile Datenverarbeitung werden drei Anwendungsbereiche unterschieden: M-Working, M-Government und M-Living.

M-Living wird dem Bereich des Alltags mit dem Zusatz „mobil“ zugerechnet. Eine Abgrenzung des privaten und mobilen Bereichs erfolgt nicht explizit, da die Grenzen zu M-Working und M-Government nicht klar zu ziehen sind. Die hier angesprochene abstrakte Ebene der Klassifikation kann entsprechend der in den Abschnitten 3.3.2 und 3.3.3 besprochenen Klassifikationen teilweise als Subklassen oder zusätzliche Dimensionen dieser Klassifikation zugeordnet werden.

Im Zusammenhang mit Begriffen wie E-Commerce und M-Commerce werden Marktbeziehungen auf übergeordneter Ebene mittels Akronymen wie z. B. B2B, B2C, B2A, C2C, C2A, B2E[265] beschrieben. Da die Einteilung aus dem E-Commerce kommt, wur-

---

[259] Vgl. Ortner 2005a, S. 45.
[260] Vgl. mik21 2004, S. 2.
[261] Vgl. Roth 2002.
[262] Vgl. Merz 2002.
[263] Vgl. Lehner 2003.
[264] Vgl. mik21 2004, S. 3.
[265] Hierbei stehen „B“ für Business, „C“ für Consumer, „A“ für Administration und „E“ für Enterprise.

de auf das Marktgeschehen fokussiert. Die Teilnehmer am Marktgeschehen werden hier bereits im Vorfeld rollenspezifisch klassifiziert und entsprechend dieser Rollen in Beziehung zueinander gesetzt.

Kuhn[266] fasst diese Beziehungen – angelehnt an die jeweilige Anbieter-Anwender-Beziehung – für mobile Anwendungen etwas weiter und fügt Beziehungen, wie Intra-business-Anwendungen, Peer-to-Peer (P2P), Machine-to-Machine (M2M) und Individual-to-Administration zu den oben genannten Akronymen hinzu. Zu fast jeder der genannten Kategorien wurden im Rahmen der von Kuhn präsentierten Studie Anwendungen diskutiert. Gegenstand der Diskussion waren jeweils Anwendungsfamilien (Anwendungen eines Nutzungsbereichs, z. B. Banking) innerhalb der Beziehungs-Kategorien. Kuhn erwähnt Anwendungsfamilien wie M-Shopping, M-Payment, M-Gaming, M-Banking, M-Voting, M-Health-Care, M-Education. Die Einteilung von Kuhn ist problemspezifisch motiviert, da er sie aus Sicht von Abrechnungsmodellen vornimmt.

Kuhn bringt Person und Individuum als Rolle ein und erläutert diese Differenzierung nicht weiter. Durch die Ergänzung der Akronyme um den Bereich M2M und P2P lassen sich in dieser Klassifikation auch neue Entwicklungen für den mobilen Bereich berücksichtigen. Die Möglichkeit zur Erweiterung des Systems um weitere Akronyme scheint gegeben. Kuhn stellt hier jedoch zwei Klassifikationen nebeneinander, welche nicht klar miteinander verbunden werden können. Beim Versuch der Einordnung von Anwendungsfamilien zu einzelnen Beziehungskategorien lässt sich jedoch leicht feststellen, dass Mehrfachzuordnungen nicht umgangen werden können (z. B. Einordnung von M-Payment).

Ob diese Klassifikation künftig als grundlegend gelten kann, darf bezweifelt werden, da sie dem Grunde nach auf die Klassifikation Verwaltungssysteme, Wirtschaftssysteme und Alltagssysteme zurückgeführt werden kann. Die Rollen von Personen und deren Beziehungen als Klassifikationsmerkmal sind aus Sicht des Marktes sicher interessant (z. B. für die Abgrenzung von Zielgruppen im Zusammenhang mit anderen Eigenschaften einer Anwendung) und können bspw. in einer synthetischen Klassifikation Berücksichtigung finden.

[266] Vgl. Kuhn 2003, S. 37f.

3.3.2 Realisierungsnahe Konzepte im mobilen Umfeld

Angelehnt an Merkmale des Ubiquitous Computing (Miniaturisierung, Einbettung, Vernetzung, Allgegenwart, Kontextabhängigkeit) gemäß Hilty et al.[267] werden verbreitete mögliche Klassifikationen aufgezeigt.

Mit dem Merkmal „Miniaturisierung“ ließe sich eine Klassifikation nach Endgeräten verbinden. So können auf Geräteebene Consumer-Endgeräte und Business-Endgeräte unterschieden werden. Weitere Klassen können diesem Ansatz entweder hierarchisch oder als zusätzliche Dimensionen hinzugefügt werden.

Schiller[268] klassifiziert nach Kommunikationsgeräten, um zu zeigen, dass Mobilität und drahtlose Kommunikation nicht das gleiche sind (siehe 3.1.3.2). Diese Einteilung entspricht auch der Klassifikation von Roth[269].

Eine andere Möglichkeit ist die Unterscheidung mobiler und drahtloser Endgeräte geordnet nach deren Leistungsfähigkeit. Somit ergeben sich die Klassen: Sensoren, integrierte Steuerungen, Pager (Rufmelder), Mobiltelefone, PDAs, Taschencomputer und Notebooks.

Die Art der Vernetzung bzw. Kontextabhängigkeit bilden ein Merkmal für mobile Anwendungen und damit auch eine Möglichkeit zur Klassifikation. Kuhn[270] differenziert (problemorientiert nach Abrechnungsmodellen) nach der Verwendung von Übertragungstechnologien. Dabei werden drei reichweitenabhängige Bereiche identifiziert: der persönliche Bereich (PAN) unter Verwendung von bspw. Infrarot oder Bluetooth, der Nahbereich (LAN) unter Verwendung von bspw. WLAN und der Fernbereich (WAN) unter Verwendung eines Mobilfunknetzes.

Von Tielert[271] werden in einem AmI-Projekt innerhalb des Szenarios „Assisted Living“ zusätzlich zur vorgenommenen qualitativen Differenzierung (medizinische Fernüberwachung, Notruf und Notfallerkennung, Leit- und Sicherheitssystem, Hausautomation und Haushaltsroboter, drahtlose Steuerung und Kommunikation) aus technologischer Sicht folgende Merkmale zur Unterscheidung identifiziert:

---

[267] Vgl. Hilty et al. 2003.
[268] Vgl. Schiller 2003.
[269] Vgl. Roth 2002.
[270] Vgl. Kuhn 2003, S. 24f.
[271] Vgl. Tielert 2004.

Arten der Vernetzung (hier eingeteilt nach Netzwerk-Reichweite):

- Funknetz am Körper (BAN),
- Funknetz im Haus (HAN),
- Funknetz im Umgebungsbereich (Multi-hop).

Einsatz von Funk- und anderen Übertragungstechnologien in diesem Zusammenhang:

- ZigBee Funknetz (als Ergänzung zu DECT, WLAN und Bluetooth),
- Induktive Nahfeldübertragung (RFID),
- Kapazitative Nahfeldübertragung,
- Ultra-Wideband (UWB),
- SmartDust – Optisches Netz.

Die Klassifikation ist problembezogen und daher als eine Auswahl von Merkmalen bzw. Kategorien aus übergeordneten Klassen zu sehen.

Für die dritte Generation von mobilen Telekommunikationssystemen klassifiziert Sang-Bum Park[272] das Umfeld von Anwendungen nach der Reichweite eingesetzter Technologien in In-Building, Urban, Suburban und Global.

3.3.3 Abstrakte Konzepte im mobilen Umfeld

Roth[273] baut auf die von ihm eingeführten Begriffe und Sichten (Benutzersicht und Sicht der Netzwerke) im Zusammenhang mit Mobile Computing verschiedene Abstraktionsebenen auf, die aus seiner Sicht als Richtlinien für eine sinnvolle Einteilung gelten können. Die beiden inneren Ringe in Abbildung 26 repräsentieren realisierungsnahe Konzepte. Im äußeren Ring sind Begriffe der höchsten Abstraktionsstufe angeordnet. Die Begriffe Ubiquitous Computing, Nomadic Computing, Personal Computing und Wearable Computing bezeichnen dabei Konzepte aus Sicht der Benutzer, ohne auf eingesetzte Technologien einzugehen. Roth räumt ein, dass *„alle Themen einer Abstraktionsstufe eine benachbarte Stufe gleichermaßen beeinflussen.*[274]*“* In seinen Ausführungen vertieft er jedoch lediglich Aspekte der realisierungsnahen Konzepte.

---

[272] Vgl. Sang-Bum Park 2004.
[273] Vgl. Roth 2002, S. 13f.
[274] Roth 2005, S. 13.

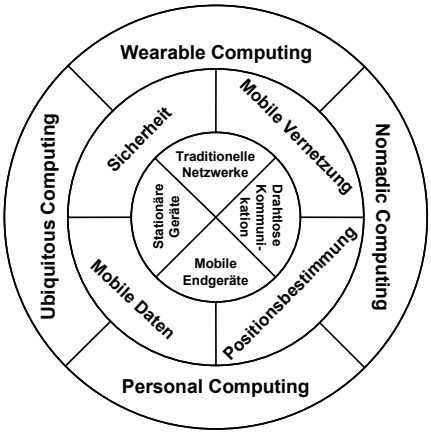

Abbildung 26: Einordnung verschiedener Begriffe im Mobile Computing[275]

Für die oberste Abstraktionsebene ist kritisch anzumerken, dass die verwendeten Begriffe bereits in einer hierarchischen Beziehung zueinander stehen. So ist z. B. Wearable Computing eine Ausprägung von Ubiquitous Computing. Die zugrunde gelegte Definition von Personal Computing nach Pandya[276] kann in diesem Zusammenhang als Hinweis auf einen privaten Bereich gesehen werden. Die Einordnung von Roth ist nicht geeignet zur Klassifikation von Anwendungsgebieten des mobilen Bereichs, da wesentliche Leistungsanforderungen an eine Klassifikation nicht erfüllt werden bzw. beschriebene Nachteile im System überwiegen. Dieser Ansatz liefert keine klare Zusammenfassung von isolierten Inhalten, keine Umgehung von Verwandtschaftsbeziehungen und der grundlegende Zugang zur Thematik wird nicht erleichtert.

Vier Dimensionen nach Lehner[277]

Lehner differenziert in seiner Klassifikation für mobile Anwendungen vier Grundklassen: Kommunikationsart, technische Basis bzw. Plattform und Dienst, allgemeine Basisfunktionen, Anwendungsbereich und Domäne.

---

[275] In Anlehnung an Roth 2002, S. 13.
[276] Vgl. Pandya 2000.
[277] Vgl. Lehner 2003, S. 16.

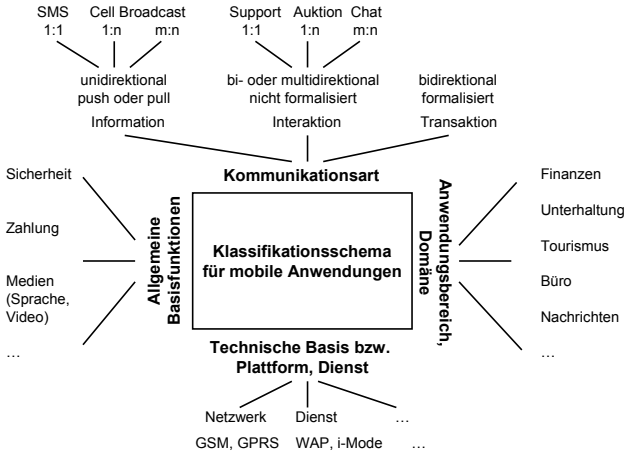

Abbildung 27: Klassifikationsschema für mobile Anwendungen[278]

Jede dieser vier Dimensionen wird zumindest beispielhaft in Subdimensionen aufgelöst (siehe Abbildung 27). Lehner will mit diesem übergeordneten Klassifikationsansatz andere, in der Literatur verbreitete, aber seines Erachtens wenig aussagefähige Einteilungen in Kategorien wie z. B. Information, Kommunikation, Transaktion und Entertainment[279] ablösen und dennoch eine allgemeine Klassifikation anbieten. Er stellt selbst fest, dass auch dieser Ansatz nicht völlig überschneidungsfrei ist, insbesondere zwischen Basisfunktionen und Anwendungsfunktionen. Auch räumt er eine gewisse Austauschbarkeit der Zuordnungen ein. So wurde bspw. die Zahlungsfunktion ursprünglich den Finanzdienstleistungen zugeordnet, während sie inzwischen als allgemeine Grundfunktion verstanden wird, da sie auch für andere Funktionen wie bspw. Hotelbuchung im Tourismus benötigt wird.

Alle Kommunikationsarten und auch alle erwähnten Technologien finden in Wirtschaft, im Verwaltungsbereich und im privaten Bereich Anwendung. Ebenso verhält es sich mit den aufgezählten allgemeinen Basisfunktionen. Die Anwendungsbereiche (Domänen) können ebenfalls alle Bereiche auf Metaebene betreffen. Positiv anzumerken ist, dass Lehner nur etablierte Begriffe in seiner Klassifikation verwendet und keine Begriffe („Modebegriffe"), für die nicht absehbar ist, ob sie eines Tages als etablierte Bezeichnung eines Anwendungsgebietes gelten können oder ob sie innerhalb kurzer Zeit wieder aus dem Sprachgebrauch des Mobile Computing verschwinden.

---

[278] In Anlehnung an Lehner 2003, S. 16.
[279] Vgl. Hansmann et al. 2001, S. 29.

Anwendungsfelder des Ubiquitous Computing nach Lipp[280]

Lipp führt aus, dass *„Ubiquitous Computing zwar ein neues, aber kein in sich abgeschlossenes Gebiet[281]“* ist. Im Gegensatz zu Roth[282] erschließt Lipp die Thematik Ubiquitous Computing aus Sicht alternativer Konzepte für Computerschnittstellen und unterscheidet zwischen Ambient Intelligence (mit Ambient Displays), Tangible User Interface (als greifbare Schnittstellen) und Augmented Reality (erweiterte „Realität“ i. S. von anreichern mit zusätzlichen Informationen) als Anwendungsfelder des Ubiquitous Computing. Für jeden der Teilbereiche können wieder Anwendungsgebiete genannt werden. Für Lipp[283] mit Bezug auf Ullmer und Ishii[284] sind bspw. Anwendungsgebiete von Tangible User Interfaces (TUI) in abstrakten Zusammenhängen Anwendungen, die auf Informationsabruf, -speicherung und -manipulation basieren. Er nennt folgende konkrete Einsatzgebiete: Lernprozesse, Unterhaltung, Visualisierung, Modellierung und Konstruktion, Systemmanagement und Konfiguration sowie Programmierung.

Für Lipp stellen die genannten Anwendungsfelder keine vollständige Aufzählung dar und er macht auch die vorhandenen Überschneidungen der Anwendungsfelder deutlich. Auf klassische Anwendungsfelder des Mobile Computing geht Lipp nicht ein, da Mobilität selbst in den Anwendungen des Ubiquitous Computing enthalten ist (siehe Abschnitt 3.1.3.3). Er beschreibt Ubiquitous Computing und seine Technologien in unmittelbarem Zusammenhang mit dem Anwender. Den genannten Einsatzgebieten lassen sich zwar Anwendungen aus dem mobilen Bereich zuordnen, als Klassifikation für diesen Bereich entsprechen sie jedoch nicht in ausreichendem Maße den geforderten Leistungsmerkmalen für Klassifikationen (siehe Abschnitt 3.2.2).

Ambient Intelligence

Für Malkewitz[285] ist Ambient Intelligence eine Komposition aus Ubiquitiy, Transparency und Intelligence im Zusammenhang mit der Unterstützung von Aktivitäten von Personen. Die Anwendungsbereiche von AmI umfassen „alles was man so tut“, wie z. B. Reisen, Erholung, Einkaufen, Arbeiten, Schlafen, Sport und Krankheit.

---

[280] Vgl. Lipp 2004.
[281] Lipp 2004, S. 113.
[282] Vgl. Roth 2002.
[283] Vgl. Lipp 2004, S. 82.
[284] Vgl. Ullmer; Ishii 2000.
[285] In Anlehnung an Malkewitz 2004.

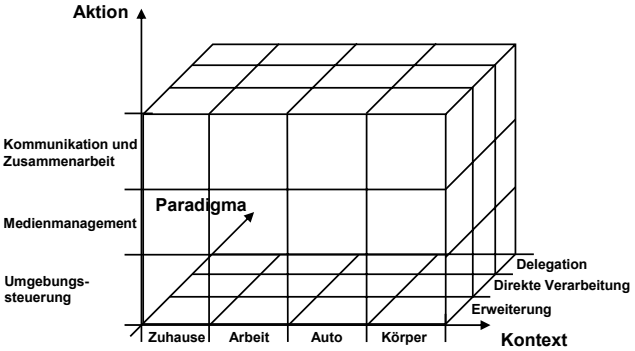

Abbildung 28: Ambient Intelligence Space[286]

Malkewitz spannt für Anwendungskontexte im Zusammenhang mit AmI einen Raum mit den Dimensionen Aktion, Kontext und Paradigma auf (siehe Abbildung 28). Damit vertieft er die Sichtweise auf Anwendungsgebiete auf nicht hierarchische Art. In der Mehrdimensionalität seiner Darstellung stellt er – im Gegensatz zu Lehner[287] und Roth[288] – keinen Zusammenhang mit realisierungsnahen Konzepten her. Diese Mehrdimensionalität stellt einen wesentlichen Vorteil des Ansatzes von Malkewitz dar. Die von ihm beispielhaft aufgezählten Anwendungsbereiche („alles was man so tut") lassen sich auf Ebene der aufgezählten Beispiele (= Anwendungen) weitgehend einordnen. Die Anwendungsbereiche selbst sind nicht direkt im Modell zu identifizieren. Klassen von Anwendungsgebieten können also erst an Hand induktiv zugeordneter Anwendungen bzw. Produkte gebildet werden. Inwieweit der aufgespannte Raum sinnvoll erweitert werden kann, entweder innerhalb einer Dimension oder um eine zusätzliche Dimension, kann hier nicht abgeschätzt werden. Definiert als „AmI-Space" ist die Einordnung von „nicht intelligenten" Anwendungen ausgeschlossen. Inwieweit sich der Raum weiter aufspannen lässt, soll an dieser Stelle nicht geklärt werden.

Die Einteilung von Malkewitz ist dem Ansatz von Tielert[289] hierarchisch untergeordnet. Es werden sukzessive Anwendungsbereiche (vergleichbar den Anwendungsfamilien nach Kuhn[290] in Abschnitt 3.3.1) für AmI erforscht und entworfen und in Anwendungsclustern zu sogenannten Szenarien zusammengefasst (z. B. Assisted Living[291]).

---

[286] Vgl. Malkewitz 2004.
[287] Vgl. Lehner 2003.
[288] Vgl. Roth 2002.
[289] Vgl. Tielert 2004.
[290] Vgl. Kuhn 2003.
[291] Vgl. Tielert 2004.

Aktuell gibt es noch wenige Szenarien. Inwieweit dieser Klassifikationsansatz für den mobilen Bereich als zielführend betrachtet werden kann, ist derzeit nicht abzuschätzen. Kriterien wie Erweiterungsfähigkeit, Zusammenfassung von isolierten Inhalten, Verbesserung der Präzision bei der Wiedergewinnung von Information können mit dieser Klassifikation erfüllt werden. Inwieweit klassische mobile Anwendungsgebiete in solche Szenarien einbezogen werden, bleibt offen. Es wäre aber auch denkbar, dass AmI als Subklasse von Anwendungsbereichen des täglichen Lebens gesehen werden kann, die Ebenen also vertauscht werden.

Matrixklassifikation

Zur Identifikation der privaten Unterstützungsbereiche eines Menschen ist eine Betrachtung der Tätigkeiten des Alltags notwendig. Die dabei gefundenen Bereiche werden als „Handlungsfelder" bezeichnet. Diese ergeben mögliche Anwendungsfelder, die eine Basis für eine Anwendungsarchitektur mobiler Anwendungssysteme ergeben. Ortner[292] unterteilt sieben Handlungsfelder (siehe Abbildung 29):

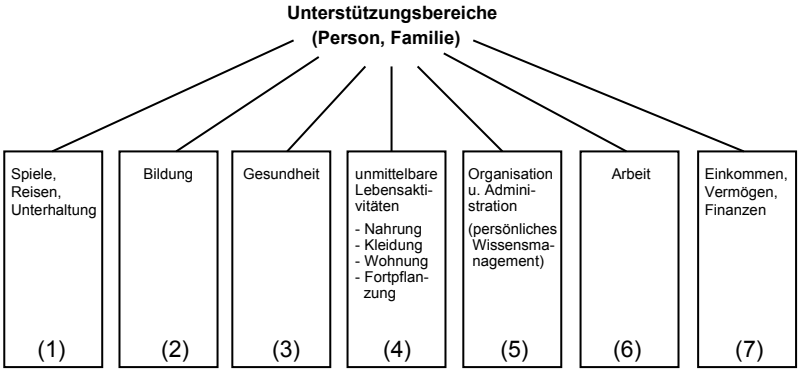

Abbildung 29: Handlungsfelder von Menschen[293]

Eine zweite Klassifikationsdimension ergibt sich durch die Berücksichtigung des Interaktionspartners. Damit wird der kommunikative Aspekt bei der Nutzung moderner Endgeräte durch mobile Anwendungen hervorgehoben.

[292] Vgl. Ortner 2005b, S. 13.
[293] In Anlehnung an Ortner 2005b, S. 13.

Analog zu der häufig verwendeten „Business to {Business, Consumer, Government}" Einteilung im Bereich des Electronic Commerce[294], können für die Interaktionspartner folgende Klassen gebildet werden:

- *Ich-Interaktion*:
  Interaktion mit dem (unterstützenden) Computer oder nur mit sich selbst. Diese Kategorie erscheint zunächst als Widerspruch zur Intention der Klassifikation. Es handelt sich aber um eine logisch zwingende Klasse. Es gibt Anwendungen, bei denen der Anwender mit niemandem – ausser sich selbst – interagiert. Auch das Denken ist – i. S. von Reden mit sich selbst – eine Interaktion.

- *Interaktion mit anderen Menschen*:
  Diese Interaktionsform bezieht sich auf die Mensch-Mensch-Interaktion und stellt damit einen weit verbreiteten Fall dar.

- *Interaktion mit Unternehmen:*
  Der Mensch kommt auch ausserhalb seiner beruflichen Tätigkeit an vielen Stellen mit Unternehmen in Kontakt. Zum einen auf Betreiben der Unternehmen hin (bspw. durch Werbung) – zum anderen aber primär aufgrund der Bedürfnisse des Menschen, z. B. beim Nutzen von Dienstleistungen.

- *Interaktion mit der öffentlichen Verwaltung:*
  Auch in der Freizeit kommt der Mensch mit Behörden und anderen öffentlichen Organen in Kontakt. Unter den Begriff „Öffentliche Verwaltung" fallen hier verschiedene öffentliche Dienstleistungen, wie bspw. Nahverkehr und Energieversorgung.

Diese zweidimensionale Klassifikation, im weiteren Matrixklassifikation (siehe Abbildung 30) genannt, umfasst Handlungsfelder und Interaktionspartner. Es werden 28 Anwendungsfelder für Unterstützungssysteme unterschieden. Dies ermöglicht eine sehr feine Einteilung des Gebietes.

[294] Vgl. Merz; Tu; Lamersdorf 1999, S. 329.

		Handlungsfelder						
		Unterhaltung, Spiele, Reisen	Bildung	Unmittelbare Lebensaktivitäten	Geld und Finanzen	Gesundheit	Persönliche Organisation	Arbeit
Interaktionspartner	Ich							
	andere Menschen							
	Unternehmen							
	Öffentliche Verwaltung							

Abbildung 30: Matrixklassifikation der Anwendungsfelder

Themen- und Anwendungskonzepte nach Graeve[295]

Bei Graeve findet sich ein umfassender Ansatz zur Klassifikation der globalen Informationsgesellschaft (siehe Abbildung 31). Sie differenziert fünf Ebenen. Top down werden ausgehend von der Vision (globale Informationsgesellschaft) vier Themenkonzepte (Digital Economy, Cyber Community, Smart Home, Individual Life Style) unterschieden.

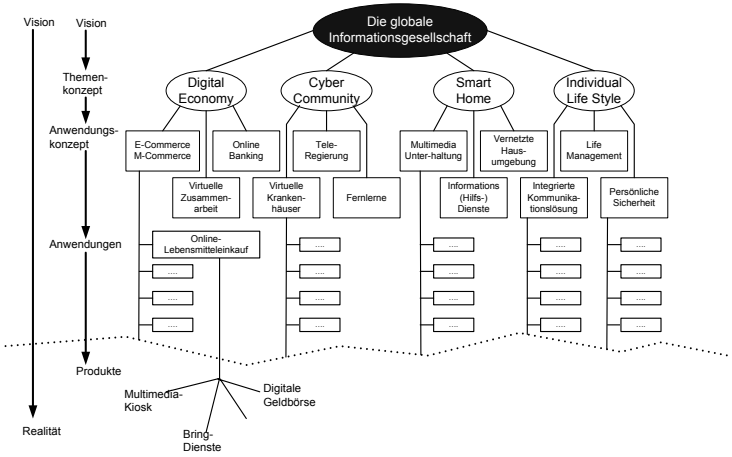

Abbildung 31: Gestaltungsfelder für die globale Informationsgesellschaft[296]

Diesen Themenkonzepten werden in den folgenden Ebenen Anwendungskonzepte und Anwendungen zugeordnet. Themenkonzepte und Anwendungskonzepte adressieren durchweg mobile Bereiche.

---

[295] Vgl. Graeve 2001.
[296] In Anlehnung an Graeve 2001.

3.3.4 Zusammenfassung

Sowohl Roth[297] als auch Turowski[298] nehmen Bezug auf Graeve und bedienen sich ihrer Klassifikation. Leider hat Graeve die Idee, welche die Grafik in Abbildung 31 repräsentiert nicht näher beschrieben. Sie spricht lediglich davon, dass die globale Informationsgesellschaft *„gekennzeichnet ist durch die allgegenwärtige Verfügbarkeit von Informationen, durch die starke Vernetzung von Lebens- und Arbeitswelten und durch das Zusammenspiel von Automatisierung und Individualisierung*[299]*“*. Der Begriff „Globale Informationsgesellschaft“ impliziert also Mobilität. Demnach lassen sich die Gestaltungsfelder für die globale Informationsgesellschaft nach Graeve auch als Gestaltungsfelder für die mobile Informationsgesellschaft verwenden, und diese können sowohl auf Ebene des Themenkonzepts als auch auf Ebene des Anwendungskonzepts gleichzeitig als Klassifikation für den mobilen Bereich dienen. Auch im Hinblick auf eine mögliche Erweiterung und Weiterentwicklung des Modells (z. B. um realisierungsnahe Merkmale in einer mehrdimensionalen Darstellung) ist keine Einschränkung erkennbar. Der Ansatz nach Graeve erscheint demnach als nützliche analytische Klassifikation für Anwendungsgebiete des mobilen Bereichs, obwohl er nicht umfassend ist.

Wie in den einzelnen Abgrenzungen und Bewertungen der dargestellten Konzepte sichtbar wird, werden bei nahezu allen Klassifikationsansätzen für Anwendungsgebiete des mobilen Bereichs Defizite identifiziert. Alle gezeigten Ansätze nehmen in ihren Einteilungen Bezug auf Mobilität, jedoch ist bei keinem der gesamte Bereich aktuell und vollständig abgebildet. Inwieweit die betrachteten Systeme der nach wie vor gegebenen Dynamik in Forschung und Entwicklung in diesem Gebiet gerecht werden können, konnte in den einzelnen Bewertungen nur vermutet und nicht fundiert belegt werden.

Zur Erschließung des Ubiquitous Computing fordert Tandler[300] die Identifikation gemeinsamer Eigenschaften von Ubiquitous Computing-Anwendungen, wobei zukünftige Entwicklungen und Erweiterungen vorherzusehen und Anforderungen aus verschiedenen Forschungsgebieten zu beachten sind. Auch Roth[301] und Lehner[302] haben

---

[297] Vgl. Roth 2002.
[298] Vgl. Turowski; Pousttchi 2004.
[299] Graeve 2001, S. 12.
[300] Vgl. Tandler 2004.
[301] Vgl. Roth 2002.
[302] Vgl. Lehner 2003.

dies zum Ausdruck gebracht. Daraus lässt sich folgern, dass eine induktive Vorgehensweise zur Einordnung von Anwendungen mit der Möglichkeit zum späteren problemspezifischen Clustering (zu Anwendungsgebieten) hilfreich sein kann, die vorhandenen Ansätze zu ergänzen oder zu vervollständigen.

Es kann daraus aber auch eine neue Klassifikation entstehen. Als Methode für eine n-dimensionale Einordnung von Anwendungen bietet sich der morphologische Kasten, der Denkpraxis Zwickys[303] folgend, an. Auf Basis von Kriterien, Eigenschaften und Aspekten aus den, in den vorhergehenden Abschnitten dargestellten Ansätzen, kann ein solcher morphologischer Kasten grundlegend strukturiert und auf dieser Basis auch weiter entwickelt werden. Eine vollständige Morphologie soll hierbei nicht aufgebaut werden (siehe Abbildung 32).

Themenkonzept nach Graeve	Digital Economy	Cyber Community	Smart Home	Individual Life Style								
Anwendungskonzept nach Graeve	…	…	…	…	…	…	Multi-media/ Unter-haltung	Informations (Hilfs-) Dienste	Vernetzte Hausum-gebung	Integrierte Kommuni-kations-lösung	Life Manage-ment	Persönliche Sicherheit
Handlungsfelder nach Ortner	Arbeit	Finanzen	Bildung	Gesundheit	Organisation	Unmittelbare Lebensaktivitäten	Unterhaltung					
Arten von Mobilität	Keine Mobilität	Benutzermobilität	Endgerätemobilität	Dienstmobilität								
Arten von Endgeräten	Sensor	Integrierte Steuerung	Pager	Mobiltelefon								
	PDA		Taschencomputer	Notebook								
Eigenschaften „heute" nach Lehner	Ortsunab-hängigkeit	Convenience	Erreich-barkeit	Sicherheit	Sofortige Verfügbarkeit							
Eigenschaften „morgen" nach Lehner	Lokalisierbarkeit	Kostengünstigkeit	Personalisierung									
Übertragungstechnologie	DECT	UWB	ZigBee	SmartDust	WLAN	Bluetooth	Nahfeldübertragung: induktiv	Nahfeldübertragung: kapazitativ				
Vernetzungstechnologie	BAN	HAN	Multi-hop	…								
Reichweite der Vernetzung nach Sang-Bum Park	In-Builing	Urban	Suburban	Global								
Interaktionspartner	Keine	Privatanwender	Unternehmen	Öffentliche Verwaltung								
Rollen und Beziehungen	C2B	B2C	C2C	C2A	A2C	M2C	C2M	M2M	P2P	…		

Abbildung 32: Morphologischer Kasten für mobile Anwendungen

Die Kategorien und deren Ausprägungen sind nicht vollständig (angeführte Ausprägungen können bei Weiterentwicklung auch Kategorien darstellen) und sollen lediglich als Ansatz dienen, einen morphologischen Kasten zu schaffen, der hilfreich ist, die deduktiven Klassifikationsansätze zu vervollständigen bzw. zu ergänzen und ggf. Erweiterungen für den gezeigten induktiven Ansatz zu identifizieren. Auch eine eigenständige Verwendung eines morphologischen Kastens zur Anwendungsklassifikation

303 Vgl. Zwicky 1966.

mit der Möglichkeit des problemspezifischen Clustering von Anwendungsgebieten für den Zielbereich ist denkbar.

Zusammengefasst kann festgestellt werden, dass die grobe Klassifikation in Abbildung 1, die Technologien und Nutzungsarten aufzeigt, aufgrund der hohen Defizite der in diesem Abschnitt vorgestellten Ansätze als bisher beständigste Klassifikation aufgefasst werden darf.

3.4 Basissystemarchitekturen

Im Folgenden werden einige Beispiele für Basissystemarchitekturen beschrieben, welche die Interaktion verteilter Anwendungen im mobilen Umfeld ermöglichen und dabei auf unterschiedliche Art und Weise Aufgaben des Anwendungsmanagements übernehmen.

3.4.1 JINI

Java Intelligent Network Infrastructure[304] (Jini) wurde Anfang 1999 von Sun Microsystems zur spontanen Vernetzung von Geräten eingeführt. Ziel der spontanen Vernetzung ist einerseits die Realisierung einer kommunikationstechnischen Verbindung von Geräten in ad-hoc Netzwerken und andererseits die Integration von durch mobile Geräte angebotenen Diensten in Dienstföderationen[305]. Mit Hilfe der Jini-Architektur können Geräte jeder Art ohne weiteres in ein bestehendes Netzwerk eingeführt werden, so dass alle anderen Komponenten unmittelbar darauf Zugriff erhalten. Ganz im Sinne der Programmiersprache Java wird hierbei versucht, das heterogene Netzwerk als eine Welt oder einen Computer darzustellen, wobei hier der Aspekt der gemeinsamen Ressourcennutzung im Vordergrund steht.

Nachdem bspw. Drucker, Speicherchips, Lautsprecher oder PDAs am Netzwerk angeschlossen wurden, erfolgt die automatische Registrierung in einem zentralen Verzeichnis. Sobald andere Benutzer oder Komponenten auf diese neuen Ressourcen zugreifen möchten, wird das für die Kommunikation notwendige Programm aus dem Netzwerk herunter geladen. Spezielle Installationen erübrigen sich, da alle relevanten Informationen wie z. B. Druckertreiber über das Netzwerkverzeichnis erhältlich sind. Insbesondere für mobile Endgeräte eröffnen sich neue Möglichkeiten, da diese prob-

[304] Sun 2006a.
[305] Vgl. Kehr 2000.

lemlos aus einem Netzwerk entnommen und in ein anderes beliebiges Netzwerk eingefügt werden können.

Das vollständig in Java implementierte Jini ist über ein Application Programming Interface (API) definiert, welches die Interaktion der verschiedenen Komponenten im Rahmen der Dienstvermittlung ermöglicht. Die Jini Architektur sieht drei Arten von Komponenten vor: Dienste, Client und den Lookup-Service.

Dienste:
Dienste können durch unterschiedliche Geräte im Netzwerk angeboten werden. In Jini werden sie als Objekte dargestellt, die über Methodenaufrufe in Anspruch genommen werden können. Die Spezifikation von Diensten erfolgt mittels Java Language Interfaces[306].

Client:
Über das zentrale Netzwerkverzeichnis (Lookup-Service) können Clients sogenannte Proxy-Objekte ausfindig machen, die für die Nutzung eines bestimmten Dienstes benötigt werden.

Lookup-Service:
Der Lookup-Service ist die zentrale Registrierungskomponente, die für die Vermittlung der Dienste notwendig ist. Dienste müssen sich hier registrieren, so dass sie von Clients gefunden werden können.

Dienstregistrierung:
Nachdem der Server bzw. der Jini-Dienst den Lookup-Service im Rahmen des sogenannten Discovery ausfindig machen konnte, eröffnet der Lookup-Service eine Verbindung zum Client (sogenanntes Callback). Über diese Verbindung wird dem Server ein Lookup-Service-Proxy, ein Objekt in serialisierter Form zugesendet, in dem die Methoden (register, lookup, notify) für die Kommunikation zwischen Server und Lookup-Service enthalten sind. Anschließend erzeugt der Server ein Service-Item-Objekt, der das Service-Proxy-Objekt und weitere Beschreibungen des Dienstes enthält und für zukünftige Clients verfügbar sein wird. Das Service-Item-Objekt wird an den Lookup-Service gesendet. Ressourcen und Dienste, die nicht mehr im Netzwerk benutzt

[306] Vgl. Zeidler; Gruteser 1999.

werden oder verfügbar sind, werden nach einer Weile automatisch vom Lookup-Service freigegeben.

Service-Lookup:
Möchte ein Client einen Service nutzen, so muss er den Lookup-Service mit Hilfe der Lookup-Methode (siehe Dienstregistrierung) kontaktieren. Daraufhin erhält der Client die Proxy-Objekte der gesuchten Dienste in serialisierter Form vom Lookup-Service zugesendet. Diese Proxy-Objekte werden von dem Client deserialisiert und für die Kommunikation mit dem Dienst verwendet. Eine ausdrückliche Spezifikation der ausgetauschten Daten ist hierdurch nicht notwendig. Es wird ausdrücklich keine Lösung für die inhaltliche Integration (Kommunikation) angeboten.

Es kann festgestellt werden, dass Jini durch die zentralisierte Dienstregistrierung und -vermittlung mit Hilfe des Lookup-Service hervorragend für nomadische Netzwerke aber weniger für ad-hoc Netzwerke geeignet ist. Es wäre zwar vorstellbar jeden Dienst mit einem eigenen Lookup-Service auszustatten, dies entspricht jedoch nicht der eigentlichen Zielsetzung. Die Bandbreiten des Netzwerks müssen den notwendigen aufwändigen Austausch von Programmcode und Daten unterstützen, was im Falle instabiler und teuerer Verbindungen als Einschränkung zu betrachten ist.

Jini erlaubt eine sehr interessante abstrakte Modellierung von Kommunikationsprotokollen, ohne explizit die ausgetauschten Daten zu spezifizieren, wie im Fall herkömmlicher Protokolle. Hierdurch wird die Realisierung von verteilten Applikationen auf den unteren Kommunikationsschichten erleichtert und die Skalierbarkeit des Frameworks entscheidend gesteigert. Fallen einzelne Dienste aus, so werden sie einfach vom Lookup-Service aus dem Verzeichnis gelöscht, so dass sie für weitere Clients nicht mehr verfügbar sind[307]. Die Problematik der inhaltlichen Integration wird auf die Anwendungsschicht verlagert.

Die Übertragung von Daten und Code vom Dienstanbieter zum Dienstnachfrager stellt auch hinsichtlich der Schnittstellenstandardisierung einen weiteren Schritt auf dem Weg zur Anwendungsintegration dar, obwohl hierbei keine inhaltliche Integration herbeigeführt wird. Vor dem Hintergrund der konfigurationsfreien Bereitstellung von Diensten im Netzwerk können bspw. zusätzliche Funktionalitäten wie grafische Schnittstellen übertragen werden, ohne dass eine Standardisierung erforderlich ist.

[307] Vgl. Zeidler; Gruteser 1999.

Die höhere Abstraktionsstufe im Bereich der Kommunikation ist entscheidend für die Überwindung der Heterogenität in verteilten Netzen, zu der Jini einen beachtlichen syntaktischen Beitrag leistet. Diese Architektur setzt allerdings voraus, dass jedes Jini-fähige Gerät mit einer Java Virtual Machine (JVM) ausgestattet ist, eine Anforderung, die für kleinste mobile Geräte derzeit noch zu hoch ist. Ansätze, die sich mit der Aufhebung dieser Einschränkung und der Einsetzung von Jini auf Klein- und Kleinstgeräte beschäftigen, können bei Preuß[308] gefunden werden.

Hinsichtlich des Sicherheitsaspektes kann die Entwicklung von Jini noch nicht als abgeschlossen betrachtet werden. Die Kommunikation mit dem Lookup-Service erfolgt über Remote Method Invocation (RMI)[309], so dass die Sicherheit der Kommunikation erst mit der Fertigstellung von „RMI Security Extension Framework“ gewährleistet werden kann. Ansätze zur Lösung dieses Problems können bei Hasselmeyer[310] gefunden werden.

Zusammenfassend kann festgestellt werden, dass Jini eine überaus interessante Infrastruktur zur syntaktisch integrierten Vermittlung von Diensten anbietet, die mittels Abstraktion erfolgreich die steigende Komplexität mobiler verteilter Anwendungen reduziert. Durch die Vermittlung von Hard- und Softwarediensten erfüllt Jini insgesamt viele der hier geforderten Anforderungen an ein Anwendungsmanagement, nicht jedoch im Hinblick auf die universelle inhaltliche (semantische) Integration der Anwendungen. Es ermöglicht die Nutzung von Rechenleistung anderer Geräte (vgl. Cyberspace und Grid Computing sowie Architekturtyp „Universalsprachenorientierung“ in Abschnitt 3.2.5).

3.4.2 J2ME

Java™ 2 Platform, Micro Edition (J2ME)[311] ist die Java Antwort auf die rasante Entwicklung im Bereich mobiler Technologie. Während J2EE für Entwicklung großer Server-Anwendungen in Unternehmen und J2SE für den Einsatz auf dem PC bzw. Notebook gedacht sind, entwickelte Sun J2ME für ressourcenbeschränkte Geräte. Die Java Plattform J2ME soll nicht nur Mobiltelefone abdecken, sondern unterschiedlich ausgestattete Kleingeräte, wie Netzwerk-Computer, PDAs und eingebettete Systeme (siehe Abbildung 33).

---

[308] Vgl. Preuß 2000.
[309] Sun 2006b.
[310] Vgl. Hasselmeyer; Kehr; Voß 2000.
[311] Sun 2006c.

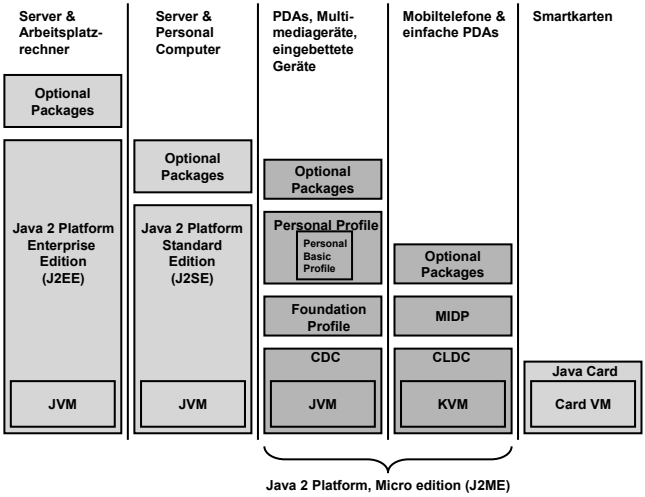

Abbildung 33: Einordnung von J2ME in die Java basierte Produktpalette[312]

Obwohl J2ME primär keine Aufgaben des Anwendungsmanagements erfüllt, stellt es eine wichtige technologische Grundlage für die Entwicklung mobiler verteilter Anwendungen dar. J2ME öffnet technologisch das Tor zur mobilen Welt für alle Java-basierten Anwendungen, die sich der kleineren, speziell für Kleingeräte entwickelten Virtual Machine (VM) bedienen können. Dadurch wird auch für diesen Bereich das Ziel verfolgt, Heterogenität von Anwendungen sowie Plattformabhängigkeiten innerhalb des Netzwerks technologisch zu überwinden, indem eine neue gemeinsame Java-basierte Plattform vorgeschlagen wird.

Ähnlich wie J2EE, J2SE und Java Card besteht auch J2ME aus einer Reihe von Spezifikationen (Java APIs) die von Experten, Geräteherstellern, Softwareproduzenten etc. definiert wurden sowie aus einer eigenen VM als Laufzeit- bzw. Ausführungsumgebung. Aufgrund seiner unterschiedlichen Einsatzgebiete unterteilt sich J2ME in weitere Untergruppen, die jedoch nicht zueinander kompatibel sind: das Mobile Information Device Profile (MIDP)[313] und die Connected, Limited Device Configuration (CLDC)[314], wobei lediglich MIDP netzwerkfähig ist.

[312] In Anlehnung an Sun o. J.
[313] Sun 2006d.
[314] Sun 2006e.

Speziell für Mobiltelefone wurde das MIDP entwickelt, das die von Sun entwickelte Kilobyte Virtual Machine (KVM) verwendet. Diese beschränkt sich auf lediglich 32 bis 512 Kbyte Arbeitsspeicher und erfüllt damit die Anforderungen kleinster mobiler Geräte. Anwendungen für MIDP, sogenannte Midlets in Anlehnung an Applets, berücksichtigen automatisch die Größe des Bildschirms und die Benutzungsschnittstelle, wodurch sich die Benutzung der gleichen Software geräteabhängig unterscheiden kann. Weiterhin bieten die Hersteller mobiler Endgeräte, wie z. B. Motorola, Sony-Ericsson, Nokia etc., unterschiedliche Laufzeitumgebungen für Midlets an, wodurch die Heterogenität verteilter Anwendungen aufrechterhalten wird.

Insgesamt kann J2ME als wichtiger Entwicklungsschritt für das Komplexitäts- und Heterogenitätsmanagement auf technologischer Ebene im Mobile Computing angesehen werden. J2ME ist damit ein Werkzeug für das Anwendungsmanagement. Durch die Verfügbarkeit der VMs auf den unterschiedlichsten Endgeräten kann technische bzw. syntaktische Interoperabilität geschaffen werden.

3.4.3 Windows CE .NET (Windows Mobile) und andere Microsoft Produkte

Microsoft platziert eine Reihe von Produkten auf dem Markt, die Funktionen des Anwendungsmanagements unterstützen könnten. Bei diesen Produken verschwimmen die Grenzen, so dass eine klare Trennung der einzelnen Produkte und ihrer Aufgabengebiete nicht immer möglich ist.

Windows Mobile[315] ist ein Betriebssystem für mobile Endgeräte, das um generische Dienste (Kalender, Email etc.) erweitert ist[316]. Mit diesem Leistungsumfang gehört Windows Mobile zu den Basissystemen. Dieses Betriebssystem ist Mitglied der Windows CE Produktreihe, deren erste Version bereits 1996 erhältlich war. Seitdem wurden daraus die Produkte Pocket PC 2000 (auch als Windows CE 3.0 bekannt), Pocket PC 2002 (Windows CE 3. 0.11171), gefolgt von Windows Mobile 2003 für Pocket PC (Windows CE 4.2) und Windows Mobile 2003 Second Edition (Windows CE 4.2.1) entwickelt. Seit Mai 2005 liegt Windows Mobile in der Version 5 (Windows CE 5.0) vor[317]. Innerhalb dieser aufgeführten Produktreihe gibt es noch spezialisierte Varianten des Betriebssystems für PDAs, Mobiltelefone (Phone Edition) und Smartphones. Die

[315] Microsoft 2006a.
[316] Vgl. Lehner 2003, S. 152.
[317] Vgl Roth 2005, S. 412-413.

Phone Edition zeichnet sich durch Telefoniefunktionen aus. Windows Mobile für Smartphones integriert Telefoniefunktionen in das Betriebssystem für PDAs.

Es werden sowohl Personal Area Networks (PAN) in Form von Bluetooth und IrDA unterstützt, wie auch Local Area Networks (LAN) mit verschiedenen Ethernet oder Token Ring Technologien und Wide Area Networks (WAN) mit Wählverbindung (engl. Dial-Up), Point-to-Point-Protocoll (PPP), Remote Access Service (RAS) und Virtual Private Networking (VPN). Damit ist das Produkt in der Lage, in stationären, in ad-hoc und in nomadischen Netzwerken eingesetzt zu werden. Somit werden synchrone und asynchrone Verbindungen unterstützt. Abhängig von der benutzten Verbindungsart und des benutzten Protokolls sind Lastverteilung und Verbindungsabbruchsicherheit integriert.

Durch die komponentenbasierte Struktur ist Windows Mobile sowohl in Bezug auf Netzwerkgröße, als auch in Bezug auf die Kombination verschiedener Netzwerke skalierbar. Generell weist eine verteilte Anwendung mit Windows Mobile und weiteren Windows-Varianten eine eindeutige Client/Server Struktur auf, schon aufgrund der verschiedenen Betriebssysteme. Es ist jedoch auch möglich, eines der Windows Mobile-Systeme Leistungen eines Servers ausführen zu lassen. Dies ist im Rahmen von Windows Mobile möglich und umfasst Serveranbindungen basierend auf Remote Access, Point-to-Point und FTP.

Ebenso verhält es sich mit der Vermittlung von Diensten. Ursprünglich basierend auf einer zentralen Dienstvermittlung, ist Windows Mobile dennoch in der Lage, auch hybride Dienstvermittlung anzubieten. Die ausgetauschten Informationen können sowohl Komponenten (z. B. durch die Technologien COM und DCOM oder mit Hilfe des sogenannten Device Management Client), als auch nur Daten sein.

Der Aspekt der Datenmodellierung wird technisch und syntaktisch ebenfalls zufrieden stellend unterstützt, da mit XML und Microsoft SQL Server zwei Technologien unterstützt werden, die eine strukturierte Datenhaltung und den Aufbau eines konzeptionellen Datenschemas ermöglichen.

Die Sicherheit eines Windows-basierten Systems wird durch verschiedene Authentifizierungsdienste (engl. Authentication Services) sichergestellt. Diese umfassen Technologien wie das Security Support Provider Interface (SSPI), NT Lan Manager (NTLM),

Kerberos, Secure Socket Layer (SSL) und Firewalls. Im Bereich der Kryptographie wird ebenfalls mit CryptoAPI eine spezifische Möglichkeit geboten.

Viel Wert wird auf die Benutzbarkeit des Systems gelegt, so dass für verschiedene Plattformen entwicklungsunterstützende Werkzeuge, sogenannte Software Development Kits (SDK), angeboten werden. Die angebotenen Werkzeuge sind für Gateways, PDAs, Mobiltelefone und Windows Thin Clients erhältlich. Das System arbeitet dabei mit den verbreitetsten Prozessortypen zusammen. Durch sogenanntes „Advanced Power Management" wird eine optimierte Energiespeichernutzung erreicht, weiterhin ist die minimale Plattform schon ab 200KB installierbar, so dass auch eine sehr effiziente Speichernutzung ermöglicht wird. Zusätzlich ist die Oberfläche in der Lage, mit Fenstern, Events und spezialisierten GUI-Elementen zu arbeiten.

Windows Mobile beinhaltet das .NET Compact Framework (aktuell Version 2.0) und unterstützt damit das aus der stationären Welt für die Entwicklung mächtiger Architekturen verbreitete .NET Framework[318].

Zusammengefasst kann man ein System bestehend aus Microsoft Windows Mobile und weiteren Microsoft-Produkten als einen weiteren Schritt in Richtung eines Anwendungsmanagements für mobile verteilte Systeme betrachten. Dieser Schritt basiert dabei auf einem erweiterten Betriebssystem und nicht auf einer Middleware. Daher handelt es sich hierbei um ein System, das nur von den Endgeräten ausgehend ein Anwendungsmanagement ermöglicht.

Über Windows Mobile hinaus bietet Microsoft weitere Ansätze, die für das Anwendungsmanagement von der technologischen und syntaktischen Seite relevant sind. In dem White Paper der Microsoft Mobile Group[319] wird das Prinzip des sogenannten Pocket PC Systems Management[320] vorgestellt, das durchaus mit dem in dieser Arbeit verwendeten Begriff des (externen) Anwendungsmanagements vergleichbar ist. Weiterhin bietet Microsoft einen Systems Management Server[321] an, der primär in stationären Windows-Systemen bereits vorhandene Möglichkeiten des internen Anwendungsmanagements (die Technologie und die Syntax betreffend) mit anderen Werkzeugen, wie Windows Management Instrumentation, Active Directory oder Windows

[318] Microsoft 2006b.
[319] Microsoft 2006c.
[320] Arildson; Dedo 2002.
[321] Microsoft 2006d.

Installer Service kombiniert. Dabei soll eine möglichst wartungsarme Umgebung mit einer unkomplizierten Installation neuer Software ermöglicht werden. In dem System werden auch verschiedene Sicherheitsaspekte berücksichtigt. Die Konfiguration der einzelnen Endgeräte wird automatisch überwacht und aktualisiert. Diese Technologie ist damit nur bedingt für den mobilen Einsatz geeignet, bietet jedoch viele der Anwendungsmanagementansätze (automatische Konfiguration, möglichst wartungsfreie Anwendungen), die andere Produkte nicht beinhalten.

Eine weitere Technologie ist der sogenannte Microsoft Mobile Information Server[322]. Dieses Produkt, das ausschliesslich für den mobilen Einsatz konzipiert ist, verfolgt im Gegensatz zu Windows Mobile einen Middleware-Ansatz, der Services über verschiedenste Transportwege anbietet und den Datenaustausch zentral steuert. Der Fokus liegt dabei auf dem Management der Daten. Dabei erreicht das Produkt eine hohe Skalierbarkeit, Lastverteilung und Überwachung der Netzaktivität.

3.4.4 CORBA

Die von der Object Management Group[323] (OMG) entwickelte Common Object Request Broker Architecture[324] (CORBA) wurde von der International Organization for Standardization[325] (ISO) zu der Standardarchitektur für verteilte Objekte (hier Komponenten genannt) gewählt. CORBA stellt ein Beispiel einer objektorientierten Middleware für stationäre verteilte Systeme – für die es auch konzipiert wurde – dar und kann somit kaum den erhöhten Anforderungen von mobilen verteilten Systemen genügen. Dennoch bietet CORBA ein Anwendungsmanagement und Lösungen, die ggf. auf die mobile Umgebung übertragen werden können.

CORBA ist eine Architektur und Spezifikation für die Erstellung, Verteilung und Verwaltung von Programmobjekten innerhalb eines Netzwerks. Im Mittelpunkt liegt die Gewährleistung der Kommunikation von Programmen, die auf unterschiedlichen Maschinen ausgeführt und von verschiedenen Unternehmen erzeugt werden. CORBA liefert ein Framework, auf dem Objekte unabhängig von der Hardwareplattform oder der Programmiersprache innerhalb eines Netzwerks mit Hilfe einer zentralen Kommunikationsschicht (Object Request Broker) kommunizieren können (siehe Abbildung 34).

---

[322] Microsoft 2006e.
[323] OMG 2006.
[324] OMG 2004.
[325] ISO 2006.

Im Unterschied zu anderen Kommunikationstechnologien basiert CORBA auf einem objektorientierten Entwurf, wobei die konkrete Implementierung mit Hilfe von Kommunikationskomponenten wie TCP/IP, Sockets, RPC etc. realisiert wird. Von solchen Details wird jedoch bei der Kommunikation zwischen Objekten abstrahiert, so dass sich der Entwickler lediglich um Schnittstellen auf einer höheren Abstraktionsebene (ab Domain-Interfaces) kümmern muss. Die Ressourcenbeschränkungen in Bandbreite und Kommunikationsleistung sind besonders zu beachten, da jeder Objektaufruf ein Netzwerkaufruf ist.

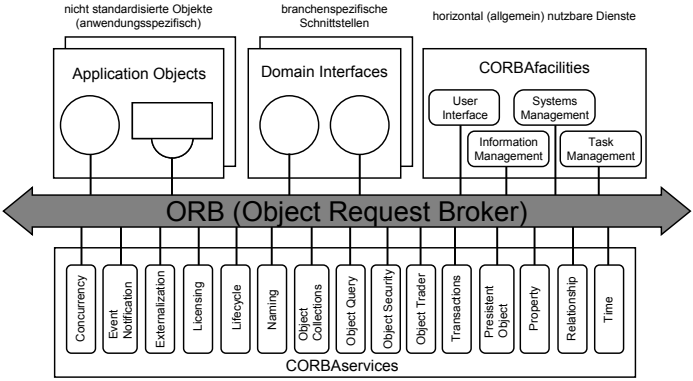

Abbildung 34: CORBA Architekturmodell[326]

Das wichtigste Konzept von CORBA ist der sogenannte zentrale ORB (Object Request Broker). Der ORB ist die Kommunikationsschicht, die sich zwischen einem CORBA-Objekt und den Benutzern des CORBA-Objektes befindet, die gleichzeitig teilweise auf dem Client und teilweise auf dem Server liegt. Der ORB ist dafür verantwortlich, Aufrufe von entfernten Objekten zu empfangen, die Objektimplementierungen festzulegen und die Kommunikation mit den Objekten zu erleichtern. Mit dem Einsatz eines Vermittlers (Broker) wird eine effiziente asynchrone Verbindung bereitgestellt. Zusammenfassend kann festgestellt werden, dass CORBA eine interessante Architektur für das Management verteilter Anwendungen definiert.

3.4.5 OpenCorba

Eine CORBA Erweiterung hin zu einer reflexiven Spezifikation stellt das von Ledoux[327] präsentierte OpenCorba dar. OpenCorba ist ein reflexiver Open Broker, der es

[326] In Anlehnung an OMG 1996, S. 9.
[327] Vgl. Ledoux 1999.

den Benutzern ermöglicht, die Darstellungs- und Ausführungsrichtlinien des Software-Busses dynamisch anzupassen. Der Begriff „Reflexion“ beinhaltet die Konzepte „Kontrolle“ und „Anpassung“. Im Rahmen der Kontrolle wird das interne Verhalten eines Systems überwacht, während die Anpassung eine dynamische Änderung dieses Verhaltens vornimmt, so dass vorhandene Eigenschaften des Systems modifiziert oder neue hinzugefügt werden können.

OpenCorba unterscheidet zwischen „class level“ (Objektebene) und „metaclass level“ (Paraebene). Klassen auf der Objektebene spezifizieren, was die Objekte solcher Klassen tun, während auf der Paraebene das Verhalten der Objekte beschrieben wird. Ein Protokoll ermöglicht, dass Eigenschaften von Objekten dynamisch zur Laufzeit verändert werden können, so dass Implementierungen angepasst und verbessert werden.

Obwohl OpenCorba viele Vorteile wie bspw. die effiziente asynchrone Verbindung und Informationsintegration durch die Einführung des „metaclass levels“ aufweist, kann es hinsichtlich der Effizienz ebenso wie CORBA nicht sinnvoll auf mobile Geräte übertragen werden. Die zusätzlichen Schichten erhöhen die Anforderungen bezüglich benötigter Ressourcen im Vergleich zu CORBA. Dennoch bietet das Prinzip der reflexiven Systeme im Zusammenhang mit dem Anwendungsmanagement ein großes Potenzial für die Reduktion der wachsenden Komplexität und Heterogenität mobiler verteilter Systeme.

3.4.6 Nexus

Das an der Universität Stuttgart am Institut für Photogrammetrie entwickelte System Nexus[328] ist eine generische Plattform zur Unterstützung von Anwendungen, die den aktuellen Aufenthaltsort eines mobilen Nutzers berücksichtigen (Location Based Services). Das Grundkonzept von Nexus ist die Repräsentation der Welt mit Hilfe von räumlichen Modellen (sogenannten Weltmodellen), welche die reale Umgebung, ergänzt durch virtuelle Objekte, beschreiben. Virtuelle Objekte werden als Vermittler zwischen der Plattform und den externen Informationsquellen eingesetzt. Für die Interaktion der mobilen Benutzer mit realen oder virtuellen Objekten wird die aktuelle Position benötigt.

Anders als die bislang vorgestellten Ansätze, die sich auf die Überwindung der technischen und syntaktischen Heterogenität in der Kommunikation konzentriert haben,

[328] Vgl. Rothermel 2005.

wurde das Nexus-System entwickelt, um Interaktion in einem heterogenen Umfeld zu ermöglichen und dies unter Nutzung des Umfeldes (Kontext) eines Anwenders mit Hilfe von Umgebungsmodellen. Die Nexus-Infrastruktur möchte dieses Problem insbesondere durch die Unterstützung des Ortsbezugs lösen. Wie in Abbildung 35 vorgestellt, gibt es vier Komponenten, die für dieses Ziel eingesetzt werden.

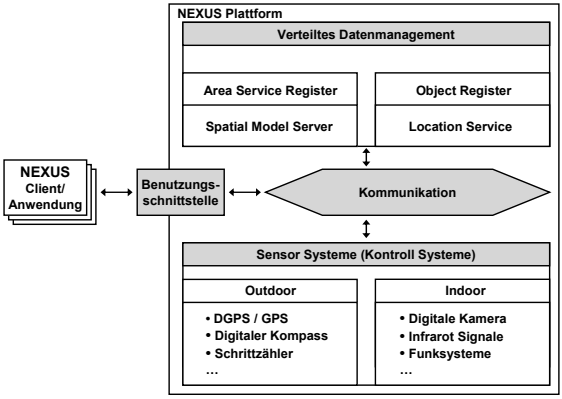

Abbildung 35: Nexus Architektur[329]

Die Komponente Benutzungsschnittstelle deckt im Wesentlichen den Benutzbarkeitsaspekt ab, indem es die grafische Darstellung und die Navigation durch das System ermöglicht. Weiterhin übernimmt sie die Anpassung an die unterschiedlichen internen Ressourcenmerkmale des Gerätes wie Rechenleistung und Speicher sowie die mögliche Qualität der Verbindung. Das Sensoren-System berücksichtigt die Tatsache, dass mobile verteilte Anwendungen sowohl in geschlossenen Räumen als auch im Freien eingesetzt werden und dementsprechend selbst die eigene Position ermitteln müssen.

Die Kommunikationskomponente berücksichtigt alle existenten Technologien, die von mobilen Kommunikationsgeräten eingesetzt werden, wie bspw. GSM, UMTS, WLAN oder Bluetooth (siehe Abschnitt 3.5.1.2). Diese Komponente hilft, technische und syntaktische Heterogenität zu überwinden, indem zwischen unterschiedlichen Kommunikationsstandards vermittelt wird. Die Umgebungsmodelle werden über das verteilte Datenmanagement bereitgestellt.

[329] In Anlehnung an Fritsch; Klinec; Volz 2000.

Obwohl hier kein Anwendungsmanagement i. e. S. (z. B. in einem fachsemantischen Sinne) erfolgt, ist der Ortsbezug ein Aspekt, der im Aufgabenbereich zukünftigen Anwendungsmanagements liegen könnte.

3.4.7 IBM's Autonomic Computing

IBM verfolgt im Rahmen seiner Forschung eine Vision unter dem Namen „autonomic computing"[330,331,332]. IBM sieht sich dabei motiviert durch die rasante Entwicklung von Technologien aus dem Bereich der Prozessoren, Speicherchips, Netzwerktechnologien, Endgeräte etc. und der damit ebenso schnell steigenden Komplexität der daraus entstehenden Computersysteme und Netzwerke (siehe „Moore' sches Gesetz, Abschnitt 1.1).

Eine Lösung dieser Komplexitätssteigerung sieht man dabei in der Erzeugung autonomer Systeme, die sich flexibel an die Umgebung anpassen, möglichst ohne Eingriff eines Menschen stabil funktionieren und sich jederzeit an die Wünsche des Nutzers anpassen. Dabei lässt man sich von der neuronalen Struktur des menschlichen Gehirns inspirieren. Ein autonomes System muss sich (seine Bestandteile) sehr genau kennen, sich ständig analysieren, optimieren, automatisch Konfigurationen ändern und anpassen sowie Installationen durchführen. Weiterhin muss es selbstheilende Fähigkeiten haben, um automatisch Lösungen für außerordentliche Situationen zu finden, und über ein durch Sensoren gesteuertes aktives „Immunsystem" verfügen, das sowohl auf Software- als auch auf Hardwareebene versucht, Gefahren (z. B. Systemabstürze) zu erkennen. Ebenfalls muss sich ein solches System an die Umgebung anpassen, in die Netzwerke der Welt einfügen und damit interagieren. Vor allem muss es all dies für den Nutzer unsichtbar tun und dabei immer dessen Nutzen optimieren.

Um diese Vision eines hoch entwickelten Anwendungsmanagements zu ermöglichen, fördert IBM die verschiedensten Forschungsprojekte, wovon sich einige im Bereich des Anwendungsmanagements bewegen. Als Beispiele seien das Projekt *Recovery Oriented Computing*[333] an der University of California in Berkeley mit dem Ziel der Entwicklung selbstheilender Systeme, das *Automizing Legacy Systems* Projekt der Columbia University mit dem Ziel der automatischen Einbindung und Nutzung von Alt-

[330] Vgl. IBM o. J.
[331] Vgl. Kephart; Chess 2003.
[332] Vgl. Boutilier et al. 2003.
[333] Vgl. Berkeley University of California o. J.

systemen und das Projekt *Astrolabe*[334] der Cornell University als Ansatz der Kontrolle und Verwaltung von automatischen Anpassungsvorgängen aufgeführt.

3.5 Technologien und (Inhalts-)Standards

In diesem Abschnitt werden Technologien und (Inhalts-)Standards vorgestellt, die zur Realisierung der Rechnerunterstützung von Aufgaben des Anwendungsmanagements beitragen. Einige der aufgeführten Technologien und (Inhalts-)Standards wurden auch bei der Implementierung des Prototypen mobiCOMP ausgetestet und eingesetzt.

3.5.1 Technologien

Im Folgenden werden einige Beispiele für Technologien vorgestellt die in mobilen verteilten Systemen vorkommen und dabei auf unterschiedliche Art und Weise für Aufgaben des Anwendungsmanagements geeignet sind.

3.5.1.1 Endgeräte

Im Umfeld des Mobile Computing gibt es eine Vielzahl unterschiedlichster Endgeräte, die Mobilität ermöglichen. Eine eindeutige und widerspruchsfreie Klassifikation ist noch nicht vorhanden, da in diesem Bereich noch viel Bewegung durch innovative Entwicklungen ist. In der Literatur finden sich einige Klassifikationsansätze, die im Folgenden überblicksartig wiedergegeben werden.

Weiser[335] und Schiller [336] verfolgen die naheliegendste Klassifikationsmöglichkeit analog zu PC-Systemen durch Einteilung nach physikalischen Eigenschaften, wie Leistungsfähigkeit (Prozessor, Hauptspeicher), Speicherplatz (permanent), Bildschirmauflösung, Bildschirmgröße, Betriebssystem, Gerätegröße, Gewicht sowie Art der Ein- und Ausgabe. Koster[337] teilt mobile Endgeräte in die vier Klassen Mobiltelefon, Notebook, PDA und Smartphone ein. Diese Einteilung berücksichtigt aber nur die populärsten Vertreter der größeren mobilen Endgeräte[338].

Alle diese Einteilungsversuche basieren auf der Grundlage des Computers als einem universell einsetzbaren Gerät. Ohne auf die Diskussion einzugehen, ob in Zukunft

---

[334] Vgl. van Renesse; Birman; Vogels 2003.
[335] Vgl. Weiser 1993, S. 77.
[336] Vgl. Schiller 2000, S. 26.
[337] Vgl. Koster 2002, S. 132.
[338] Beide Klassifikationsansätze werden in einem Beitrag von Andreou et al.2002 vergleichend zusammengefasst.

Computer Universalgeräte bleiben oder auf bestimmte Funktionen (Anwendungen) spezialisierte Geräte vorherrschen werden, ist heute feststellbar, dass mobile Endgeräte zu Gunsten einer Optimierung auf den jeweiligen Anwendungskontext hin entwickelt werden[339].

Aus diesem Grund soll eine Klassifikation der Endgeräte nicht nur auf physikalische Unterscheidungsmerkmale hin erfolgen, sondern weitere Kategorisierungsmöglichkeiten berücksichtigen, wie z. B nach der Art der Kommunikationsmöglichkeiten mit dem Benutzer, der Zweckbestimmung, dem Grad der Aufmerksamkeit oder gestalterischen Merkmalen.

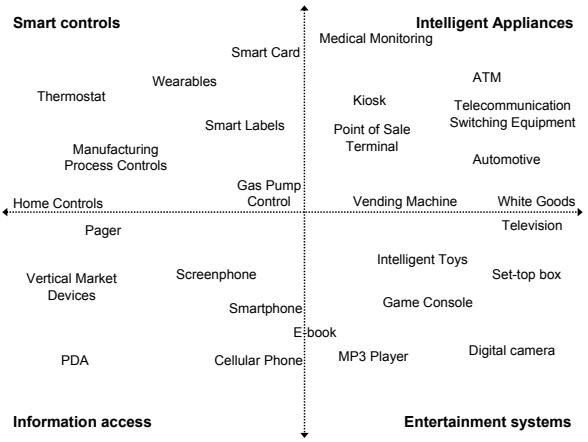

Abbildung 36: Klassifikation mobiler Endgeräte[340]

Eine grobe Aufteilung nach der Art der Zweckbestimmung verfolgen Hansmann et al.[341], indem sie die Geräte in die vier Klassen „Smart controls“, „Intelligent Appliances“, „Information access“ und „Entertainment systems“ einteilen (siehe Abbildung 36).

3.5.1.2 Kommunikationstechnik

In Anlehnung an Tanenbaum[342] können bei Kommunikationsnetzen die räumlichen Distanzen zwischen den einzelnen Endgeräten als Unterscheidungskriterium herange-

[339] Vgl. Müller-Wilken 2002, S. 19.
[340] In Anlehnung an Hansmann et al. 2001, S. 29.
[341] Vgl. Hansmann et al. 2001, S. 28.
[342] Vgl. Tanenbaum 1998, S. 25.

zogen werden (siehe Tabelle 2 aus Abschnitt 3.1.3.2). Bei drahtloser Kommunikation haben sich die Begriffe Mobiltelefonie für WAN, drahtlose lokale Netzwerke (WLAN) für LAN und Wireless Personal Area Networks (WPAN) für PAN durchgesetzt[343]. Orthogonal zu dieser Unterscheidung nach räumlicher Funkabdeckung wird im Bereich drahtloser Netzwerke noch zwischen ad-hoc Netzwerken und Infrastrukturnetzwerken unterschieden[344]. Während bei Infrastrukturnetzwerken eine feste (bestenfalls auch noch administrierte) Infrastruktur von Netzzugangsknoten erwartet werden kann, stellen ad-hoc Netzwerke keine zentralen Mechanismen für die Vernetzung bereit. Jeder aktive Netzknoten ist somit für die Kommunikation eigenverantwortlich[345].

Wireless Personal Area Network – PAN

Bluetooth[346]

Bluetooth ermöglicht eine drahtlose, serielle funkbasierte Verbindung zwischen heterogenen Endgeräten wie PCs, Mobiltelefonen, PDAs und digitalen Kameras ohne Sichtkontakt. Dieser in 1990 verabschiedete Hersteller-Standard[347] ist seit 2002 in die IEEE 802-Familie als IEEE 802.15.1[348] Standard aufgenommen worden. Bluetooth arbeitet in einem Kurzstreckenbereich (funkklassenabhängige Reichweite von 10 bis maximal 100 m) mit einem Netzwerk das aus maximal 8 aktiven Endgeräten in einem sogenannten Piconet besteht. Dieses Netzwerk kann aber um weitere Teilnehmer zu einem sogenannten Scatternet erweitert werden, indem ein Endgerät (Master) als Verbindung zweier Piconets dient. Zusätzlich können Bluetooth Protokolle sowohl asynchrone als auch synchrone Verbindung unterstützen. Bluetooth wird z. B. für Anwendungen wie Audioübertragung, Videoübertragung, Internetverbindung oder Drucken eingesetzt. Die unterschiedlichen Datenverbindungen werden als Bluetooth-Profile angeboten. Ein Hauptziel der Spezifikation ist der Ersatz herkömmlicher Kabelverbindungen. Hiermit sollte ein Beitrag zur Interoperabilität geschaffen werden. Aktuell liegt Bluetooth in der Version 2.1 vor, die eine Datenübertragungsrate von 2,1 Mbit/s ermöglicht. Eine breitbandige Version 3.0 (bis zu 480 Mbit/s) ist in Vorbereitung.

---

[343] Vgl. Roth 2005, S. 26-27.
[344] Vgl. Höpfner; Türker; König-Ries 2005, S. 60 und 62.
[345] Die Grundlagen drahtloser Netze werden bei Lehner 2003, S. 17-23, Roth 2005, S. 15-39 und Tanenbaum 2003, S. 326-353 detailliert abgehandelt.
[346] Weiterführende Literatur zu Bluetooth findet sich in Höpfner; Türker; König-Ries 2005, S: 27-29, Mutschler; Specht 2004, S. 38-40 und Hansmann et al. 2001, S. 279-288.
[347] Vgl. Bluetooth SIG 2003.
[348] IEEE 2006a.

In den meisten Fällen wird Bluetooth zum Aufbau von ad-hoc Netzwerken verwendet. Für den Einsatz von Bluetooth spricht, dass dabei nur ein geringer Energieverbrauch entsteht, die Technologie weltweit einsatzfähig ist, und durch die Rundumabstrahlung der eingesetzten Geräte eine hohe Flexibilität erreicht wird. Bluetooth sendet im lizenzfreien 2,4 GHz-Band. Dies bietet einen Kostenvorteil. Ein Problem dabei ist jedoch die mögliche Interferenz mit Geräten, die das gleiche Frequenzband nutzen, wie z. B. Mikrowellen oder WLAN.

Zur Gewährleistung von Vertraulichkeit und Integrität bietet Bluetooth Maßnahmen zur wechselseitigen Authentifikation, zur Datenverschlüsselung und zur Autorisierung. Diese Maßnahmen beruhen allein auf Gerätebasis und sind somit nicht für einzelne Dienste oder Anwender einsetzbar[349].

Infrarot [350]

Das Hauptziel von Infrarot-Kommunikation ist die drahtlose Kommunikation zwischen Geräten in Sichtweite, die normalerweise leitungsgebunden kommunizieren. Beschränkt wird diese Kommunikationsform allerdings dadurch, dass die kommunizierenden Geräte nur einen maximalen Winkel von 30 Grad zueinander haben dürfen und auch nur höchstens einen Meter voneinander entfernt sein dürfen. Werden diese Vorgaben jedoch eingehalten, lassen sich mit dieser Technologie schnell kleine Datenmengen austauschen, wie z. B. elektronische Visitenkarten oder eine Kalendersynchronisation. Die Infrarot-Kommunikation basiert auf dem 1994 veröffentlichten Firmenstandard Infrared Data Association (IrDA)[351], der eine Datenübertragungsrate von 115,2 Kbit/s ermöglicht. Aktuelle Entwicklungen im Bereich der Infrarot-Kommunikation sind in IrDA Version 1.1 zusammengefasst. Dort werden drei Geschwindigkeitsklassen angeboten: Mid-Infrared (1,152 Mbit/s), Fast-Infrared (4 Mbit/s) und Very-Fast-Infrared (16 Mbit/s).

Bei Infrarot-Kommunikation sind keine speziellen Sicherheitsmaßnahmen vorgesehen, da sich die Kommunikation aufgrund der technologischen Beschränkungen durch den Anwender selbst kontrollieren lässt und somit die meisten Angriffe verhindert werden können[352].

---

[349] Vgl. Eckert 2004, S. 854.
[350] Weiterführende Literatur zu Infrarot findet sich in Mutschler; Specht 2004, S. 40 und Hansmann et al. 2001, S. 288-292.
[351] Infrared Data Association 2006.
[352] Vgl. Eckert 2004, S. 847.

Drahtlose lokale Netzwerke – LAN

WLAN [353]

Wireless Local Area Network (WLAN) ist eine drahtlose, lokale Netzwerktechnologie für stationäre und mobile Endgeräte. WLAN basiert auf den Varianten des IEEE 802.11-Standards, einem weiteren Vertreter der LAN-Standards innerhalb der IEEE 802-Familie[354]. Diese Technik ermöglicht mobiles Arbeiten ohne Verkabelung, so dass sich der Nutzer in einem Umkreis von durchschnittlich 30 m in Gebäuden bzw. 300 m im Freien seines Anschlusses (Hotspot) nahezu beliebig bewegen kann[355]. Die am weitesten verbreiteten WLAN-Standards sind die Varianten IEEE 802.11b, a und g, mit einer Datenübertragungsrate von 11 Mbit/s (802.11b) bzw. 54 Mbit/s (802.11a und g).

WLAN stellt ein Sicherheitsrisiko dar, da durch die Funkübertragung kein gesicherter Kommunikationskanal vorliegt, wie dies bei der drahtgebundenen Übertragung der Fall ist. Eine gewisse Vertraulichkeit bietet ein Kabel aufgrund der Abschirmung der Leitungen. Bei WLAN müssen spezielle kryptografische Maßnahmen zum Erreichen der üblichen Schutzziele (siehe Abschnitt 2.5.1.4) vorgenommen werden[356].

MAN

Im Bereich der Metropolitan Area Networks gibt es Varianten mit flächendeckender WLAN-Hotspot-Versorgung durch einzelne Anbieter, aber auch gemeinsam genutze Ressourcen von (Heim-)Anwendern, die lokale WLAN-Ressourcen zur Verfügung stellen[357], wie bspw. Opennet[358].

Eine weitere geeignete und schon bereitstehende Technologie für die Netzabdeckung im Stadtbereich ist Worldwide Interoperability for Microwave Access (WIMAX)[359],

---

[353] Weiterführende Literatur zu WLAN findet sich in Höpfner; Türker; König-Ries 2005, S. 24-27 und Mutschler; Specht 2004, S. 32-35. Weitere drahtlose lokale Netzwerktechnologien, wie bspw. HiperLAN, HiperACCESS, HiperLINK und HomeRF finden sich in Mutschler; Specht 2004, S. 40-41 und Roth 2005, S: 103-106.
[354] IEEE 1990a.
[355] Vgl. Roth 2005, S. 85.
[356] Vgl. Eckert 2004, S. 806.
[357] Vgl. Wiecker 2002, S. 437.
[358] Opennet-Initiative 2006.
[359] Vgl. Mutschler; Specht 2004, S. 35-36.

ebenfalls ein Vertreter der IEEE 802-Familie (IEEE 802.16[360]), der breitbandige Funkübertragung bei einer Reichweite von bis zu ca. 3 km ermöglicht.

Mobiltelefonie – WAN

GSM [361]

Das Global System for Mobile Communication (GSM) wurde 1982 bis 1991 als europäischer Mobilfunkstandard entwickelt und kam 1992 in einigen europäischen Ländern erstmals zum Einsatz[362]. Mittlerweile hat dieser Standard globale Bedeutung. Das GSM-Netz ermöglicht außer der Sprach- auch eine Datenübertragung mit 9,6 Kbit/s[363]. Diese Datenübertragung kann über ein im mobilen Endgerät integriertes Modem genutzt werden. Durch den Einsatz des High-Speed-Circuit-Switched-Data-Protokolls (HSCSD), das mehrere Kanäle zur Übertragung bündelt, können bis zu 57,6 Kbit/s erreicht werden[364]. Das Netzwerk bietet neben reiner Datenübertragung auch Datendienste, wie Short Message Service (SMS) oder Multimedia Message Service (MMS) an. Darüber hinaus gibt es Zusatzdienste, wie bspw. die Übertragung der Rufnummer und Anrufweiterleitung. Der Zugang zum GSM-System erfolgt über eine eindeutig identifizierbare Chipkarte, dem Subscriber Identity Module (SIM), das in dem jeweiligen Endgerät installiert ist.

GPRS

Eine Erweiterung von GSM stellt der General Packet Radio Service (GPRS) dar. GPRS nutzt die Paketvermittlung zur Übertragung, während GSM Leitungsvermittlung verwendet. Mit dieser Änderung wurde der zunehmenden Datenübertragung Rechnung getragen, da leitungsvermittelnde Übertragung für Datenübertragungen nicht so gut geeignet ist wie für Sprachübertragungen. GPRS bietet den mobilen Endgeräten eine Anbindung an das Internet, so dass jedes Endgerät eine IP-Adresse erhält. Mit GPRS wird ein „always on“, also ein ständiger Zugang zum Internet, ermöglicht, bei dem der Nutzer nur für das übertragene Datenvolumen bezahlen muss. GPRS bie-

---

[360] IEEE 2006b.
[361] Weiterführende Literatur zum Themengebiet GSM findet sich in Höpfner; Türker; König-Ries 2005, S. 29-32 und Mutschler; Specht 2004, S. 25-32.
[362] Vgl. Lehner 2003, S. 32.
[363] Vgl. Roth 2005, S. 48-49.
[364] Vgl. Roth 2005, S. 64.

tet abhängig von der eingesetzten Fehlerkorrektur und der Netzauslastung Datenraten im Bereich von 48 Kbit/s bis zu 107,2 Kbit/s[365].

EDGE

Enhanced Date Rates for GSM Evolution (EDGE) reizt die GSM-Technologie weiter aus, indem die Modulation der Trägerfrequenz stark erhöht wird. Die theoretisch mögliche Bandbreite liegt bei 384 Kbit/s, wird aber in der Praxis nicht erreicht, da das Signal wegen der Überladung sehr störanfällig wird[366]. Gerade bei größeren Entfernungen zur Basisstation sinkt die Übertragungsrate aufgrund der erhöhten Fehlerrate.

UMTS [367]

Das Universal Mobile Telecommunication System (UMTS) ist ein paketorientierter Mobilfunkstandard der sogenannten dritten Generation. UMTS bietet gegenüber dem GSM-Netz mit GPRS weiter fortgeschrittene Leistungsmerkmale. Die maximale theoretische Datenübertragungsrate beträgt 2 Mbit/s im stationären Fall. In der Praxis wird die Übertragungsrate durch eine steigende Anzahl von Teilnehmern innerhalb einer Zelle gesenkt, sodass eine durchschnittliche Übertragungsgeschwindigkeit von 384 Kbit/s zu erwarten ist. Durch diese Leistungsverbesserungen und den erreichten Bandbreitengewinn werden mit UMTS neue Dienste möglich, wie bspw. Audio- oder Videostreaming[368].

Die UMTS-Architektur bietet zum Erreichen der Schutzziele Vertraulichkeit, Authentifizierung, Integrität und Nichtabstreitbarkeit wechselseitige Authentifikation der Kommunikationspartner. Das Roaming in das GSM-Netz, also die nahtlose Kommunikation mit diesem (Interoperabilität), ist von der deutschen Regulierungsbehörde für Telekommunikation und Post (RegTP) vorgeschrieben. Durch Versorgungslücken des UMTS-Netzes bestehen aber Probleme bei der Verfügbarkeit[369].

SAN

Als weltumspannendes Netzwerk (abgesehen von den Polgebieten) ohne terrestrische Stationen kann das Inmarsat-Satellitensystem zur Sprach- und Datenübertragung eingesetzt werden. Es erlaubt bidirektionale Datenübertragung mit ISDN-Geschwindig-

[365] Vgl. Wiecker 2002, S. 431-432.
[366] Lehner 2003, S. 51-54.
[367] Weiterführende Literatur zu UMTS findet sich in Höpfner; Türker; König-Ries 2005, S. 32-34.
[368] Vgl. Wiecker 2002, S. 433f.
[369] Vgl. Mattern 2003, S. 106.

keit (64 Kbit/s). Man kann zwischen leitungs- und paketorientierten Verbindungen wählen. Inmarsat wird hauptsächlich in Regionen mit fehlender oder schwach ausgeprägter (terrestrischer) Infrastruktur (z. B. im Schiffsverkehr) eingesetzt[370].

3.5.1.3 Service Discovery Protocol (Bluetooth)

Für das Dienstmanagement in einem ad-hoc Netzwerk beinhaltet Bluetooth eine Spezifikation namens Service Discovery Protocol[371] (SDP). Die Dienste eines jeden Bluetooth-Gerätes werden hierbei bei einem SDP-Server des Piconets registriert. Ein SDP-Server verwaltet eine Menge von „Service Records“, welche die Dienste repräsentieren. Server wird dabei jeweils das Endgerät, das die erste Verbindung hergestellt hat. Die SDP-Clients können über den Server nach Diensten suchen und auf diese zugreifen.

Obwohl Bluetooth aufgrund seiner stark begrenzten Netzwerkgröße[372] unter dem Aspekt der Skalierbarkeit Mängel aufweist, sind viele der genannten Anforderungen des Anwendungsmanagements in ad-hoc Netzwerken realisiert. Das Netzwerk ist sehr flexibel, passt sich dynamisch an, kann mit Bandbreitenunterschieden und Verbindungsabbrüchen umgehen und bietet eine interessante Möglichkeit des Dienstmanagements. Im Bereich der ad-hoc Netzwerke ist Bluetooth als Protokoll und Technologie damit durchaus zu einem Vorreiter im Bereich des Anwendungsmanagements zu zählen.

3.5.1.4 TCP/IP

Im Jahr 1969 wurden die ersten vier Rechner über eine längere Distanz zum ARPAnet[373] vernetzt. Mit diesem Netzwerk wurde das Ziel verfolgt, weltweit verteilte Rechner miteinander zu verbinden, das auch dann noch funktioniert, wenn Teile des Netzwerkes ausgefallen sind. Mit der Entwicklung des Transmission Control Protocol/Internet Protocols (TCP/IP) im Jahr 1974, wurde das bis heute vorherrschende Protokoll zur Vernetzung heterogener Rechnerplattformen geschaffen. Nach einer Reihe von Anpassungen und Designänderungen stellte das ARPAnet am 01.01.1983 sein Netzwerkprotokoll auf TCP/IP um. Damit war das Internet geschaffen. Seitdem ist TCP/IP der offene paketbasierte Datenübertragungsstandard für das Internet[374].

---

[370] Vgl. Inmarsat o. J.
[371] Bluetooth SIG 1999.
[372] Maximal 8 Teilnehmer spannen ein Piconet auf, mit Hilfe von Scatternets kann die Anzahl der Netzwerkteilnehmer erhöht werden.
[373] ARPAnet war ein Netzwerk der Advanced Research Projects Agency (ARPA), einer amerikanischen militärischen Forschungseinrichtung.
[374] Vgl. Tanenbaum 1998, S. 70-71.

3.5.1.5 Web Services

Das World Wide Web Consortium (W3C) definiert Web Services mit einem konkreten Bezug auf die XML-Technologie wie folgt: *„A Web service is a software application identified by an Uniform Ressource Identifier (URI), whose interfaces and bindings are capable of being defined, described, and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.*[375]*"*

Die Funktionalitäten von Web Services reichen vom einfachen Nachfragen von Informationen bis hin zu komplexen Geschäftsprozessen. Ein Web Service wird eingesetzt, um andere Anwendungen bzw. Web Services zu finden und diese einzubeziehen. Dabei verhält sich ein Web Service wie eine „Black-Box"-Komponente (siehe Abschnitt 4.2.1.3). Die Ein- und Ausgaben haben eine bekannte Spezifikation. Die Komponente bietet eine Funktionalität an, ohne ihre Realisierung sichtbar zu machen. Einen Web Service kann man beliebig wieder verwenden und in Anwendungssysteme einbinden.

Die Architektur der Web Services basiert auf drei Bausteinen[376]: Service Broker, Service Provider und Service Requester. Der Service Broker spielt die Rolle eines Vermittlers zwischen dem Service Provider und dem Service Requester. Der Service Provider stellt dabei Dienste zur Verfügung, indem er deren Verfügbarkeit publiziert und auf Anfragen eines Service Requesters reagiert. Bekommt der Service Requester vom Service Broker eine Referenz auf den gesuchten Dienst zurück, so bindet sich der Service Requester an den Service Provider, um diesen Dienst zu nutzen. Dazu ist es notwendig, dass ein Nachfrager den Web Service findet, den er für seine Aufgabe braucht, z. B. mit Hilfe von Verzeichnissen.

In Zusammenhang mit Web Services wird man mit verschiedenen Begriffen und Standards konfrontiert. Nachfolgende Abbildung gibt einen Überblick über die grundlegenden Basissoftware-Standards und deren Sprachschicht[377]. Diese vorgestellte Schichtung wird häufig auch als Web Service Protokollstapel (engl. Web Service Protocol Stack) bezeichnet.

---

[375] Austin; Barbir; Garg 2002.
[376] Vgl. Fletcher; Waterhouse 2002.
[377] Vgl. Fletcher; Waterhouse 2002.

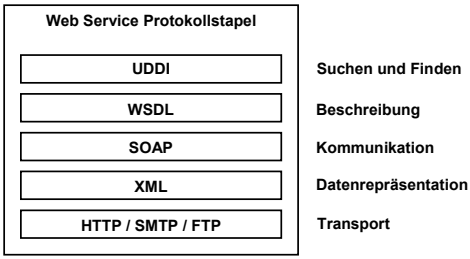

Abbildung 37: Web Services – Basissoftware-Standards[378]

Es folgt ein kurzer Überblick dieser einzelnen Basissoftware Standards[379].

- *UDDI – Suchen und Finden eines Web Services*
  Universal Description, Discovery and Integration[380] (UDDI) entspricht einem globalen Verzeichnis (Registry), welches die verschiedenen Geschäftstätigkeiten im Internet und die Informationen für die einzelnen Schnittstellen enthält, um mit den Diensten in Kontakt zu treten. UDDI ist vergleichbar mit einem Branchenverzeichnis, in dem jeder Dienstanbieter seine Angebote ablegen und in dem jeder Dienstsuchende den entsprechenden Anbieter finden kann.

- *WSDL – Vorbereiten einer Web Service Nutzung*
  Die Web Service Definition Language[381] (WSDL) liefert eine XML-basierte Schnittstellenbeschreibung und alle notwendigen Informationen, um einen Web Service aufzurufen. Hiermit können zwei Anwendungen miteinander integriert werden.

- *SOAP – Aufruf eines Web Services*
  Für den Aufruf eines Web Services wird das Simple Object Access Protocol[382] (SOAP) als Kommunikationsprotokoll benötigt. Die zu übermittelnden Daten werden in einer XML-Repräsentation, wie in einen Umschlag, „verpackt“, um zwischen den Anwendungen versendet werden zu können.

---

[378] In Anlehnung an Fletcher; Waterhouse 2002.
[379] Vgl. Westphal 2002, S. 33.
[380] OASIS 2006.
[381] W3C 2001.
[382] W3C 2004.

- *XML – Datenrepräsentation*
  Die Extensible Markup Language[383] (XML, siehe Abschnitt 3.5.2.2) ist eine vom W3C standardisierte Auszeichnungssprache zur Beschreibung und zum Austausch von Daten. Dabei ist XML eine Metasprache, die eine Strukturierung der Daten und Dokumente und eine Beschreibung ihrer Eigenschaften ermöglicht. Durch die einheitliche Struktur (Syntax) sichert XML den Datenaustausch zwischen Web Service Anwendungen.

- *HTTP/SMTP/FTP – Transport Protokolle*
  Für den Transport der XML-Daten werden die Standard Internet Transportprotokolle, wie Hypertext Transfer Protocol[384] (HTTP), Simple Mail Transfer Protocol[385] (SMTP) und File Transfer Protocol[386] (FTP) verwendet.

3.5.1.6 Coda und Odyssey

Ein für große nomadische Netzwerke konzipiertes System ist das Dateisystem Coda[387], das über zwei wichtige komplementäre Mechanismen zur Datensynchronisation verfügt. Durch die sogenannte Serverreplizierung werden Kopien einer Datei auf mehreren verschiedenen Rechnern gespeichert. Der zweite Mechanismus besteht aus einem zusätzlichen Betriebsmodus, in dem anstatt replizierter Speicherseiten auf der Client-Seite zuvor zwischengespeicherte (gecachte) Daten verwendet werden.

Coda setzt voraus, dass dem Client sicherere und nicht ausfallgefährdete Server zur Verfügung stehen. Die Anforderung einer stationären Kerninfrastruktur ist im Zusammenhang mit mobilen verteilten Systemen als hoch anzusehen. Obwohl mit dem zweiten Betriebsmodus auch Verbindungsabbrüche berücksichtigt werden, gelten sie jedoch eher als Ausnahmen. Hierdurch kann es vorkommen, dass mobile Endgeräte häufig mit zwischengespeicherten (redundanten) und ggf. nicht mehr aktuellen Daten arbeiten müssen.

Als Nachfolger von Coda wurde das Odyssey-System eingeführt, das zusätzlich Kontextbezug und anwendungsspezifisches Verhalten ermöglicht. Auch hier wird von no-

[383] W3C 2006a.
[384] W3C 2005.
[385] IETF; Postel 1982.
[386] IETF; Postel; Reynolds 1985.
[387] Vgl. Satyanarayanan et al. 1990.

madischen zentralisierten Netzwerken ausgegangen, die über einen stationären und vertrauenswürdigen Server verfügen[388].

Anwendungen melden benötigte Ressourcen und den Grad der erforderlichen Verfügbarkeit beim Odyssey-System an, das dann diese Werte entsprechend überwacht. Fällt die aktuelle Verfügbarkeit einer Ressource unter einem bestimmten Wert, so wird die Anwendung von dieser Veränderung unterrichtet, so dass sie sich entsprechend den neuen Bedingungen anpassen kann.

Obwohl Odyssey für das mobile Umfeld besser geeignet ist, als der Vorgänger Coda, besitzt es auch einige Nachteile. Hierzu zählen insbesondere die Nichtberücksichtigung eingeschränkter Ressourcen auf mobilen Endgeräten und der unzuverlässigen und meist teuren Verbindungen, da Odyssey die Übermittlung vollständiger Sammlungen von Dateien erfordert.

3.5.1.7 Bayou

Das von Xerox PARC entwickelte Bayou Projekt[389] liefert eine Infrastruktur, welche die Interaktion verschiedener Anwendungen im mobilen Umfeld insbesondere jedoch in nomadischen Netzwerken ermöglicht. Das Bayou-System ist eine Plattform von replizierten und konsistenten mobilen Datenbanken. Unter Berücksichtigung der instabilen Konnektivität, die das mobile Umfeld charakterisiert, verwaltet dieses System die Inkonsistenz von Daten, die als Folge von simultanen Zugriffen eintreten kann.

Das Erkennen und Beseitigen von Inkonsistenzen der Daten erfolgt unter Einbezug der betroffenen Anwendungen, da Datenkonflikte direkt von der durch die Anwendung festgelegten Semantik abhängen. Das Bayou-System erlaubt Anwendungen, eine eigene anwendungsspezifische Definition von „Konflikt“ festzulegen, die dann vom System implementiert und verfolgt wird. Hierdurch wird die automatische Erkennung solcher Konflikte möglich (engl. automatic detection). Im Falle einer Inkonsistenz erfolgt die automatische Behandlung durch Zusammenführung der Daten (engl. merge procedure). Dies gelingt i. d. R. ohne Beteiligung des Anwenders. Durch die einheitliche Handhabung von Konflikterkennung und -beseitigung kann die Konsistenz aller Systeme zugesichert werden.

---

[388] Vgl. Capra; Emmerich; Mascolo 2002, S. 43.
[389] Vgl. Terry et al. 1995.

Obwohl Bayou fortschrittlicher hinsichtlich der Erkennung und Behandlung von Inkonsistenzen ist als Coda und Odyssey, hat auch Bayou den Nachteil einer Client/Server Architektur, die für ad-hoc Anwendungen ungeeignet ist. Das System erfordert die vollständige Replikation jeder Datensammlung auf mehrere Server, wodurch ressourcenarme mobile Endgeräte, wie z. B. Handhelds lediglich die Rolle des Clients übernehmen können.

3.5.1.8 Xmiddle

Für das Problem der Nachbildung und des Transfers einer hohen Anzahl von Dateien bietet die am Department of Computer Science des University College London entwickelte datenaustauschorientierte Middleware Xmiddle[390] einen guten Lösungsansatz. Das Xmiddle Projekt richtet sich speziell an die Bedürfnisse mobiler Endgeräte in ad-hoc Netzwerken und ermöglicht den Austausch von Daten im XML-Format für Anwendungen auf verschiedenen Endgeräten. Aufgrund der XML-Basierung resultierte „Xmiddle“ als Projektname.

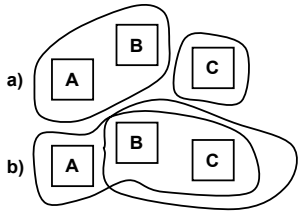

Abbildung 38: Datenübertragung mit Xmiddle[391]

Die Datenübertragung erfolgt während des Online-Betriebs. Die Replizierung und Aktualisierung von Daten erfolgt im Offline-Betrieb, so dass erst nach Wiederherstellung der Verbindung ein Abgleich ermöglicht wird. Um den Umfang der übertragenen Daten zu minimieren, benutzt Xmiddle eine spezielle Baumstruktur, die gezielt Teile des Baumes und nicht die komplette Dateiensammlung austauscht. Abbildung 38 zeigt wie beim Übergang vom Offline- (a) zum Online-Betrieb (b) nur ein Teil des Datenbaumes auf dem Host B mit dem Datenbaum auf dem Host C abgeglichen wird. Hierdurch wird insbesondere das Problem des übertragenen Datenumfangs gelöst, das bei Coda und Odyssey eine große Schwachstelle darstellt[392].

---

[390] Vgl. Mascolo et al. 2002.
[391] In Anlehnung an Capra; Emmerich; Mascolo 2002, S. 46.
[392] Vgl. Capra; Emmerich; Mascolo 2002, S. 47.

3.5.1.9 Tuple-Spaces: Lime, Tspaces und JavaSpaces

Tuple-Spaces basieren auf einem vom mobilen Endgerät und einem Server gemeinsam genutzten Datenraum, in dem beliebig Daten geschrieben und gelesen werden können[393]. Mit „Server“ muss hier nicht notwendigerweise ein stationärer Rechner gemeint sein, sondern es kann sich auch um ein Gerät beliebiger Art handeln, das in diesem Moment Serverfunktionalität ausübt. Existiert eine Verbindung zwischen diesem Server und dem Client, dann benutzen beide den gleichen virtuellen Datenraum und können asynchron Daten austauschen. Bricht die Verbindung ab, bleiben die lokalen Datenräume weiter vorhanden, sie werden jedoch nicht mehr ausgetauscht. Somit funktionieren beide Systeme weiter. Hierbei muss gewährleistet sein, dass die Daten bei einer neuen Verbindung synchronisiert und Datenkonflikte beseitigt werden.

Ein weiterer Vorteil für den Einsatz im mobilen Umfeld ist, dass unabhängig von der Technologie oder Plattform der kommunizierenden Geräte, eine Zusammenarbeit alleine auf dem Tuple-Space möglich wird. Die Heterogenität der verschiedenen Endgeräte wird damit wirksam überwunden.

Diese Technologie ist für den Informationsaustausch in ad-hoc Netzwerken geeignet und bietet insbesondere Vorteile in abbruchgefährdeten Bereichen oder bei stark wechselnder Bandbreite. Die Technologie arbeitet asynchron, ist jedoch zu keiner Lastverteilung fähig.

Ein Nachteil ist jedoch der fehlende Kontextbezug. Da die Systeme nicht merken sollen, wenn keine Verbindung existiert, ist bei Tuple-Space-Systemen auch kein Kontextbezug implementiert. Ein solcher Tuple-Space-Ansatz könnte in Verbindung mit einem Zwischensprachen-Konzept den fehlenden Kontextbezug herstellen. Auch der Austausch von Komponenten und Diensten ist möglich.

Es existieren mehrere Systeme für Tuple-Spaces. Zu nennen sind Linda[394] und die Erweiterung Lime[395], die nicht nur einen Tuple-Space, sondern auch sogenannte Projektionen[396] auf diesen Speicherraum ermöglicht. Tspaces[397] von IBM ist eine Middlewa-

---

[393] Vgl. Gelernter 1985.
[394] Vgl. Gelernter 1985.
[395] Vgl. Picco; Murphy; Roman 1999.
[396] Eine Projektion (Attributbeschränkung) extrahiert einzelne Attribute aus der ursprünglichen Attributmenge und kann somit als eine Selektion auf Spaltenebene verstanden werden.
[397] Vgl. Wyckoff et al. 1998.

re, die Java-Technologie, Tuple-Spaces und Datenbanktechnologie vereinigt. Es verwendet eine klare Client/Server-Struktur und bietet dabei eine Funktionalität an, welche die Mächtigkeit einer relationalen Datenbank erreicht. Da nur die Server Tuple-Spaces besitzen und die Clients nur minimalen Zugriff haben, ist dieses System jedoch nicht mehr im ad-hoc Bereich, sondern nur noch in nomadischen Netzwerken einsetzbar. JavaSpaces[398] benutzt Objekte als Tupel, kann jedoch ebenfalls nur in nomadischen Netzwerken eingesetzt werden.

Zusammengefasst handelt es sich hier um eine Technologie, die aufgrund ihrer Flexibilität für das dynamische und heterogene Umfeld mobiler Technologien geeignet ist und in Verbindung mit einem Zwischensprachen-Konzept im Bereich des Anwendungsmanagements genutzt werden kann, um Kommunikation und Verwaltung zwischen verteilten Anwendungen zu schaffen.

3.5.2 Meta- und objektsprachliche (Inhalts-)Standards

Im Folgenden werden einige Beispiele für Meta- und objektsprachliche (Inhalts-)Standards vorgestellt, die auf unterschiedliche Art und Weise für Aufgaben des Anwendungsmanagements in mobilen verteilten Systemen geeignet sind.

3.5.2.1 HTML

Die Hypertext Markup Language[399] (HTML) ist eine textbasierte Auszeichnungssprache zur Darstellung (Form) von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt. Neben den vom Browser angezeigten Inhalten einer Webseite enthält HTML auch Metainformationen, die den Inhalt (Semantik) einer Webseite beschreiben oder Informationen über die Sprache, Autoren oder Erstellungsdatum bereitstellen. Die Auszeichnungssprache wurde vom World Wide Web Consortium[400] (W3C) bis Version 4.01 weiterentwickelt.

Zur Adressierung anderer Dokumente im Internet werden innerhalb des Dokumentes Hyperlinks verwendet. Dies ist die Grundlage für das World Wide Web. Namengebend sind die Hypertext-Elemente, die zum Verweis auf andere Textstellen oder auf ein anderes Dokument dienen.

---

[398] Vgl. Waldo 1998.
[399] W3C 2006b.
[400] W3C 2006c.

Dem Text wird durch Auszeichnung (engl. markup) von Textteilen mit in der Regel paarweisen (öffnenden und schließenden) Tags eine Struktur (Syntax) verliehen. Ein Tag beginnt immer mit dem Zeichen „<“. Es folgt eine entsprechende Buchstabenfolge und, wenn keine Attribute verwendet werden, endet ein Tag mit einem „>“. Die jeweils zusammengehörenden Tags bilden zusammen mit dem dazwischenliegenden Text (Inhalt) ein Element. Diese Elemente lassen sich nach Regeln, die in einer Dokumenttypdefinition[401] (DTD) angegeben sind, verschachteln:

```
<p>Ein Textabsatz, der ein <em>betontes</em> Wort enthält.</p>
```

Mit HTML wird eine beschreibende und keine verfahrens- bzw. darstellungsorientierte Textauszeichnung erreicht. Das heißt, HTML-Tags sind keine Angaben zur Präsentation, die dem Webbrowser mitteilen, wie er den Text (zwischen den Tags) visuell zu formatieren hat. Vielmehr sind Tags eine strukturierende Auszeichnung, mit der sich Textbereiche eine Bedeutung zuordnen lassen, z. B. `<h1>…</h1>` für eine Überschrift, `<p>…</p>` für einen Textabsatz und `<em>…</em>` für einen betonten Text. Wie diese Bedeutung letztlich dem Benutzer vermittelt wird (im Falle einer Überschrift z. B. durch vergrößerte, fette Schrift), ist zunächst dem Webbrowser überlassen und hängt von der Ausgabe-Umgebung ab. Obwohl HTML-Dokumente in der Regel auf Computerbildschirmen dargestellt werden, können sie auch auf anderen Medien ausgeben werden, etwa auf Papier oder mittels Sprachausgabe.

3.5.2.2 XML

Die Extensible Markup Language[402] (XML, engl. für erweiterbare Auszeichnungssprache) ist ein Standard, der vom World Wide Web Consortium (W3C) definiert wurde. XML definiert dabei Regeln für den Aufbau halbstrukturierter Daten in Form einer Baumstruktur. Dabei enthalten XML-Dokumente Daten, die zum Teil einer fest vorgegebenen Struktur entsprechen, teilweise aber auch Elemente beinhalten, die keinem statischen Schema entsprechen. XML lässt als Rahmenkonzept offen, ob und wie ein konkretes XML-Dokument automatisiert verarbeitet werden kann.

Für ein XML-verarbeitendes Programm (XML-Anwendung) müssen die Elemente der jeweiligen Dokumente genau beschrieben werden. Dies betrifft insbesondere die Festlegung der Strukturelemente und ihre Anordnung innerhalb des Dokumentenbaums.

---

[401] W3C 2006a.
[402] W3C 2006a.

Auch für diese Beschreibung selbst stellt XML Standards zur Verfügung, wie bspw. die einfache DTD bzw. XML Schema[403].

XML ist damit ein Standard zur Datenrepräsentation und eine Metasprache zur Definition von beliebigen, in ihrer Grundstruktur jedoch stark verwandten Auszeichnungssprachen, die Programme ebenso wie Daten beschreiben können[404]. Die Namen der Strukturelemente (XML-Elemente) für eine XML-Anwendung lassen sich frei wählen. Ein XML-Element kann ganz unterschiedliche Daten enthalten und beschreiben: meistens Text, aber auch Grafiken oder abstraktes Wissen. Ein Grundgedanke der XML ist, Daten und ihre Repräsentation zu trennen, um Daten bspw. einmal als Tabelle und einmal als Grafik auszugeben, aber für beide Arten der Auswertung die gleiche Datenbasis im XML-Format zu nutzen.

XML-Dokumente besitzen einen physischen und einen logischen Aufbau. Der physische Aufbau eines XML-Dokuments besteht aus Entitäten. Die erste Entität ist die Hauptdatei des XML-Dokuments. Weitere mögliche Entitäten sind über Referenzen (`&name` für das Dokument bzw. `%name` für die Dokumenttypdefinition) eingebundene Zeichenketten, evtl. auch ganze Dateien sowie Referenzen auf Zeichenentitäten zur Einbindung einzelner Zeichen, die über ihre Nummer referenziert werden (`&#Dezimalzahl` oder `&#xHexadezimalzahl`). Eine XML-Deklaration wird optional verwendet, um XML-Version, Zeichenkodierung und Verarbeitbarkeit ohne Dokumenttypdefinition zu spezifizieren. Eine Dokumenttypdefinition wird optional verwendet, um Entitäten sowie den erlaubten logischen Aufbau zu spezifizieren.

Der logische Aufbau eines XML-Dokumentes ist eine Baumstruktur und damit hierarchisch strukturiert. Als Baumknoten gibt es:

- Elemente, deren physische Auszeichnung mittels eines passenden Paares aus Start-Tag (`<Tag-Name>`) und End-Tag (`</Tag-Name>`) oder einem Empty-Element-Tag (`<Tag-Name/>`) erfolgen kann,
- Attribute als bei einem Start-Tag oder Empty-Element-Tag geschriebene Schlüsselwort-Werte-Paare (`Attribut-Name="Attribut-Wert"`) für Zusatz-Informationen über Elemente,

[403] W3C 2006d.
[404] Vgl. Merz 2002, S. 201.

- Verarbeitungsanweisungen (`<?Ziel-Name Parameter ?>`, engl. Processing Instruction),
- Kommentare (`<!-- Kommentar-Text -->`),
- Text, welcher als normaler Text oder in Form eines CDATA-Abschnitts (`<![CDATA[ beliebiger Text]]>`) auftreten kann.

Ein XML-Dokument muss genau ein Element auf der obersten Ebene enthalten. Unterhalb von diesem Dokumentelement können weitere Elemente verschachtelt werden.

Zur Spezifikation des logischen Aufbaus werden die Dokumenttypdefinitionen zunehmend durch das umfangreichere XML-Schema abgelöst, welches keine Möglichkeit zur Definition von Entitäten, aber als Ersatz die Definition von Datentypen ermöglicht.

Einige Webbrowser können XML-Dokumente mit Hilfe eines eingebauten XML-Parsers direkt darstellen. Dies geschieht in Verbindung mit einem Stylesheet. Diese Transformation kann die Daten in ein komplett anderes Format umwandeln, das Zielformat muss dabei nicht unbedingt XML sein.

3.5.2.3 XSLT

XSL Transformations (XSLT) ist Teil der Extensible Stylesheet Language[405] (XSL). XSLT ist eine Programmiersprache zur Formulierung von Transformationsregeln von XML-Dokumenten. XSLT basiert auf der logischen Baumstruktur eines XML-Dokuments und erlaubt die Definition von Umwandlungsregeln. XSLT-Programme, sogenannte XSLT-Stylesheets, sind dabei ebenfalls nach den Regeln des XML-Standards aufgebaut.

Spezielle XSLT-Prozessoren lesen XSLT-Stylesheets ein und transformieren ein oder mehrere XML-Dokumente nach den Stylesheet-Regeln in das gewünschte Ausgabeformat. Eine Transformation besteht aus einer Reihe von einzelnen Transformationsregeln, die Templates (dt. Schablonen) genannt werden. Ein Template besitzt ein auf XPath[406] basierendes Pattern (dt. Muster), das beschreibt, für welche Knoten das Template anzuwenden ist, und einen Inhalt, der bestimmt, wie das Template seinen Teil des Zielbaums erzeugt.

---

[405] W3C 2006e.
[406] W3C 1999.

3.5.2.4 WML

Wireless Markup Language[407] (WML) ist eine XML-basierte Seitenbeschreibungssprache, die eine stark reduzierte Fassung von XHTML[408] darstellt[409]. Sie ist Teil des Wireless Application Protocols (WAP) und zur Darstellung veränderlicher Inhalte auf Mobiltelefonen entwickelt worden. WML genügt den Restriktionen mobiler Endgeräte im Hinblick auf Bandbreite, die beschränkte Darstellungsfläche, den Speicherbedarf und die Rechenleistung. Der WML-Code wird in einen speziellen Binärcode transformiert, der an das Mobiltelefon gesendet wird. Bei älteren Mobiltelefonen mit wenig Speicherplatz ist die Größe einer WML-Datei auf 1600 Byte beschränkt, inzwischen spielt das aber kaum noch eine Rolle.

WML verwendet als prozedurale Skriptsprache WMLScript[410], eine vereinfachte Version von JavaScript, zur Erweiterung der standardmäßigen Präsentationsmöglichkeiten. WML basiert auf der Kartenstapel-Metapher[411]. Hierbei gilt eine übertragene WML-Datei (gesamte Website) als Kartenstapel (engl. Decks) und die einzelnen Seiten als Karten (engl. Card). Ein Kartenstapel wird mit dem Tag `<wml>…</wml>` eingeleitet und abgeschlossen. Für jede Karte wird der Tag `<card>...</card>` verwendet. Bei der Datenübertragung wird vom Server immer ein gesamter Kartenstapel (WML-Datei) über das WAP-Gateway an das Endgerät gesendet. Mit WML werden Textrepräsentation und Layout sowie Navigation und Verlinkung von Karten (analog zu HTML) spezifiziert. Durch Parametrisierung wird eine effiziente Netzwerknutzung erreicht, indem Zeichenketten bei der Anzeige durch Variablen ersetzt werden können.

3.5.2.5 WAP

WAP (Wireless Application Protocol)[412] wurde 1997 von Ericsson, Motorola, Nokia und Unwired Planet entworfen und eingeführt. Es ist eine offene, standardisierte Architektur und Sammlung von Protokollen, die mobilen Nutzern ermöglicht, mit drahtlosen Endgeräten auf Informationen und Dienste zuzugreifen. WAP kann auf vielen mobilen Endgeräten wie Mobiltelefonen, Smartphones und PDAs eingesetzt werden und ist so entworfen, dass es in den meisten drahtlosen Netzwerken arbeiten kann[413].

---

[407] OMA 2001a.
[408] W3C 2006b.
[409] Vgl. Lehner 2003, S. 194.
[410] OMA 1998.
[411] Vgl. Rischpater 2000, S. 224.
[412] OMA 2001b.
[413] WAP Forum o. J.

Aufgrund seiner Komplexität und des anfänglich hohen Preises wurde WAP von den Kunden jedoch nicht sofort angenommen.

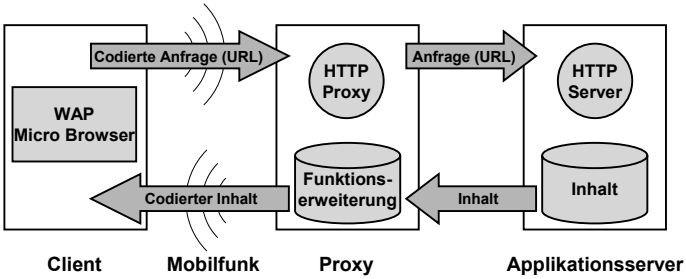

Abbildung 39: WAP-Architektur[414]

Die WAP-Spezifikation bietet viele Vorteile und Optimierungen bezüglich Ressourceneinsparungen, Leistungserhöhung oder Sicherheit, die für das Anwendungsmanagement nutzbar sind. Es wird ein kompakter Protokollstapel zur Minimierung der Bandbreitenanforderungen angeboten. Dieser garantiert auch, dass WAP-Anwendungen in verschiedensten mobilen Netzwerken genutzt werden können (siehe Abbildung 39).

WAP nutzt das sogenannte Wireless Transaction Protocol[415] (WTP), um zuverlässigen Datentransport in einem unzuverlässigen Netzwerk zu ermöglichen. WTP beinhaltet viele Optimierungen für mobile Netzwerke. Die genutzten Verbesserungen im WAP-Protokollstapel führen zu bedeutenden Einsparung in der Bandbreite. WML (Wireless Markup Language) ist besser für mobile Endgeräte geeignet als HTML (siehe Abschnitt 3.5.2.4). Die WAP-Spezifikation bietet das Wireless Transport Layer Security Protokoll (WTLS)[416], das auf dem Secure Socket Layer (SSL)[417] basiert und Datenintegritäts-, Geheimnis- und Authentifizierungsmöglichkeiten beinhaltet. Vor allem im Bereich der Sicherheit, aber auch bei der Optimierung der Bandbreitennutzung bietet WAP trotz seines für solche Protokolle hohen Alters einige interessante Ansätze.

3.5.2.6 GI-Spezifikationsrahmen für Fachkomponenten

Mit dem Spezifikationsrahmen für Fachkomponenten der GI wird ein methodischer Standard angestrebt, der die im Rahmen der Spezifikation von Fachkomponenten ein-

[414] WAP Forum o. J.
[415] OMA 2001c.
[416] OMA 2001d.
[417] Vgl. IETF; Dierks; Rescorla 2006.

zusetzenden Notationen festlegt. Um Fachkomponenten (verschiedener Hersteller) mit geringem Aufwand zu einem kundenindividuellen Anwendungssystem zu integrieren, bedarf es der Etablierung eines solchen inhaltlich-funktionalen Standards[418].

Unter der Spezifikation einer Fachkomponente verstehen Ackermann et al. eine *„vollständige, widerspruchsfreie und eindeutige Beschreibung ihrer Außensicht […], d. h. die Spezifikation zeigt auf, welche Dienste eine Fachkomponente in welchem Bedingungsrahmen bereitstellt.*[419]*“* Hierzu werden Fachkomponenten in sieben Beschreibungsebenen unterteilt, die als verschiedene Blickwinkel (Aspekte) auf die Spezifikationsobjekte zu verstehen sind (siehe Abbildung 40).

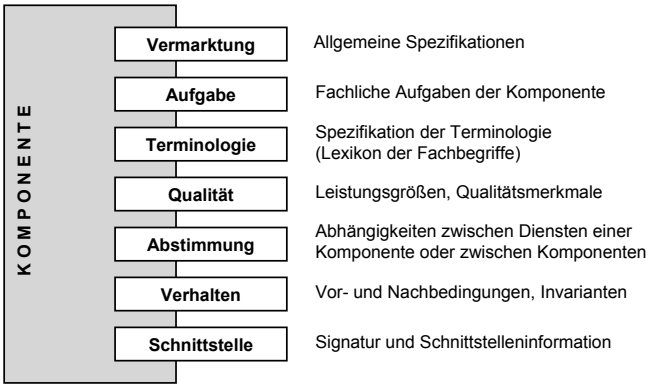

Abbildung 40: Beschreibungsebenen von Fachkomponenten[420]

Die *Vermarktungsebene* beinhaltet im Wesentlichen drei Kategorien von Daten: grundlegende Informationen über abzulegende Komponenten (Komponentenname und -version, Informationen über Bezugsmodalitäten, wie bspw. Komponentenpreis, Kontaktdaten und Vertragsbedingungen), klassifizierende Daten, welche die hierarchische Kategorisierung nach Wirtschaftszweig und Anwendungsbereich (Domäne) ermöglichen sowie Anforderungen an die zugrunde liegende Systemarchitektur (zu untergliedern in Hardware, Software und Basissysteme).

Auf der *Aufgabenebene* wird die Funktionalität der zu beschreibenden Komponente normsprachlich, d. h. basierend auf einer exakt spezifizierten Terminologie sowie

[418] Overhage 2002, S. 7.
[419] Ackermann et al. 2002, S. 10.
[420] In Anlehnung an Ackermann et al. 2002, S. 4.

normierten Satzbauplänen, hierarchisch beschrieben. Aufgaben können demnach sowohl abstraktiv als auch kompositiv beliebig in Teilaufgaben unterteilt und von entsprechenden Prozeduren der Schnittstelle referenziert werden.

Die *Terminologieebene* erfasst die von den jeweiligen Komponenten verwendeten Fachbegriffe (Termini), wobei zwischen globalen und lokalen Begriffen zu unterscheiden ist. Globale Termini werden von mehreren Komponenten gleichzeitig verwendet und ermöglichen die flexible Integration von Komponenten bei der Administration ihrer semantischen Heterogenität auf Metasprachebene (Überwindung semantischer Heterogenität durch Sicherstellung der Interoperabilität). Lokale Termini hingegen gehören zu einer einzelnen Komponente und müssen i. d. R. explizit an den Komponentenschnittstellen übersetzt werden.

Die *Qualitätsebene* erlaubt die hierarchische Speicherung von Qualitätsklassen mit zugeordneten Qualitätseigenschaften, welche wahlweise für eine gesamte Komponente (bspw. Qualitätsprädikate unabhängiger Kontrollinstitutionen oder Überdeckungsmaße) oder für einzelne Komponentenfunktionen (bspw. Durchsatz, Fehlerrate) gelten kann. In beiden Fällen werden Qualitätsausprägungen gespeichert, welche sich zwecks Vergleichbarkeit explizit auf im System vorgegebene Referenzumgebungen beziehen.

Die *Abstimmungsebene* hat zum Ziel, die Reihenfolgebeziehungen zwischen den Funktionen der Komponenten in formaler Notation (z. B. TemporalOCL) zu spezifizieren.

Die *Verhaltensebene* spezifiziert in formaler Notation (z. B. OCL) die Vor- und Nachbedingungen sowie Invarianten der Komponenten.

Die *Schnittstellenebene* umfasst neben der direkt von Programmierumgebungen verwendbaren Schnittstellenspezifikation der Komponente in Interface Definition Language (IDL) die Zuordnung von IDL-Funktionen zu Aufgaben der Aufgabenebene sowie von IDL-Datenobjekten zu Termini der Terminologieebene, wodurch semantische Korrektheit gewährleistet werden kann. Eine IDL-Funktion kann hierbei mehreren Aufgaben zugeordnet werden, ein Datenobjekt kann jedoch nur mit einem Terminus übereinstimmen.

Die der Abstimmungs-, Verhaltens- und Schnittstellenebene zugehörigen formalen Sprachen (TemporalOCL, OCL und IDL) werden in den Abschnitten 3.5.2.9, 3.5.2.8 und 3.5.2.7 näher erläutert.

Eine Weiterentwicklung dieses Spezifikationsrahmens wurde von Overhage[421] mit der Unified Specification of Components (UnSCom) erreicht.

3.5.2.7 IDL – Interface Definition Language

Die Interface Definition Language[422] (IDL) ist eine formale Notation zur Beschreibung der Schnittstelle einer Komponente. Es handelt sich dabei um eine von der Object Management Group (OMG) vorgeschlagene Sprache, welche in Wirtschaft und Forschung Akzeptanz gefunden hat. Eine IDL-Schnittstellenbeschreibung liefert dem Nachfrager einer Komponente Informationen, die er zur Verwendung der Funktionen (Dienste) benötigt. IDL ist keine Programmiersprache, sondern nur eine Sprache zur Beschreibung von Schnittstellen. In vielen Programmierumgebungen werden aber Transformationsregeln (engl. Mappings) zu IDL zur Verfügung gestellt. Dateien, welche IDL-Schnittstellenbeschreibungen enthalten, haben die Dateiendung „.idl“. Im Folgenden wird zunächst eine Definition der OMG IDL Grammatik in einer Variante der Extended Backus-Naur Form[423] (EBNF) vorgestellt (siehe Tabelle 4) und danach ein Beispiel für IDL gegeben.

Symbol	Semantik
::=	Ist definiert als
\| | Alternative |
<text> | Variable |
“text” | Literal / Terminalsymbol |
* | Die vorangestellte syntaktische Einheit kann 0,1,2,... mal wiederholt werden. |
+ | Die vorangestellte syntaktische Einheit kann 1,2,3,... mal wiederholt werden. |
{} | Die in geschweifte Klammern eingeschlossenen syntaktischen Einheiten werden zu einer syntaktischen Einheit zusammengefasst. |
[] | Die in eckige Klammern eingeschlossene syntaktische Einheit ist optional. |

Tabelle 4: Symbole der OMG IDL

[421] Vgl. Overhage 2006.
[422] OMG 2002.
[423] ISO 1996.

Eine IDL-Datei beginnt mit dem Schlüsselwort `interface und` daran anschließend der Name der Komponente. Durch diese Benennung werden untergeordnete Teile der Komponente eindeutig identifizierbar. Anschließend folgen Typdefinitionen (Schlüsselwort `typedef`) und Strukturdefinitionen (Schlüsselwort `struct`) basierend auf vordefinierten Datentypen. Diese Definitionen werden für die Spezifikation der Signaturen der Funktionen benötigt. Ausnahmezustände werden mit dem Schlüsselwort `exception` definiert. Die angebotenen Dienste (Funktionen) werden mit Eingangsparametern und dem Datentyp ihres Rückgabewerts definiert. Der Name einer Funktion dient zur eindeutigen Identifikation innerhalb der Fachkomponente. Zusätzlich zur Deklaration interner Dienste ist die Deklaration solcher Dienste, d. h. externer Dienste die von der Komponente genutzt werden, möglich.

Im Folgenden wird ein Beispiel für eine IDL-Datei vorgestellt:

```
interface Bestandskontenverwaltung {
typedef string KontoNr;
typedef double Bestand;
  struct Zeitpunkt { ... };
  struct Konto {
    KontoNr     n;
    Bestand     Sicherheitsbestand;
    Bestand     Meldebestand;
    ...};
  struct Buchungssatz { ... };
  exception ZuGeringerBestand {};
  ...
Bestand BerechneBestandZum(in KontoNr n, zum Zeitpunkt z);
...};
```

Eine umfassende Beschreibung der Syntax und Semantik von IDL findet sich in der Spezifikation von IDL[424].

3.5.2.8 OCL – Object Constraint Language

Die Object Constraint Language[425] (OCL) ist eine Sprache zur formalen Spezifikation von Invarianten (Integritätsbedingungen) von Komponenten sowie Vor- und Nachbedingungen von Komponenten-Funktionen. OCL ermöglicht somit die Beschreibung des Laufzeitverhaltens von Komponenten-Funktionen.

Invarianten sind Bedingungen, die immer erfüllt sein müssen. Eine Invariante wird durch einen Ausdruck beschrieben, dessen Auswertung „true" ergibt wenn die Invari-

[424] OMG 2002.
[425] OMG 2003.

ante erfüllt ist. Eine Vorbedingung einer Funktion ist eine Restriktion, die in dem Moment vor Ausführung der Funktion erfüllt sein muss. Eine Nachbedingung ist eine Restriktion, die in dem Moment nach der Ausführung der Funktion wahr sein muss. Jede Vor-/Nachbedingung und Invariante hat einen Kontext (Ko-Text), auf welchen sie sich bezieht (z. B. Bestandskontenverwaltung). Elementare Konstrukte von OCL sind die Schlüsselwörter `forAll`, `exists` und `iterate`. Auf `forAll` folgt ein Boolescher Ausdruck, der für alle Elemente einer Kollektion wahr sein muss. `Exists` besagt, dass mindestens ein Element der Kollektion der Bedingung genügen muss. `Iterate` bewirkt, dass über alle Elemente, die bestimmten Bedingungen genügen, iteriert und eine Ergebnisvariable aufsummiert wird. Es wird zwischen drei Arten von Kollektionen unterschieden: Sets, Bags und Sequenzen.

OCL ist keine Programmiersprache, sondern eine reine Ausdruckssprache, mit ihr wird keine Ausführungslogik beschrieben. Es handelt sich um eine typisierte Sprache, d. h. jeder OCL-Ausdruck hat einen Typ, und alle OCL-Ausdrücke müssen typkonform verwendet werden. OCL hat genauso wie IDL den Vorteil, unabhängig von einer Programmiersprache zu sein.

Im Folgenden wird ein Beispiel für eine OCL-Datei wiedergegeben:

```
Bestandskontenverwaltung
self.Konto->forAll(k:Konto | k.Sicherheitsbestand >= 0)
self.Konto->forAll(k:Konto | k.Meldebestand >=
k.Sicherheitsbestand)

Bestandskontenverwaltung::
BerechneBestandZum(n:KontoNr,z:Zeitpunkt):Bestand
pre : self.Konto->exists(k:Konto | k.KontoNr = n)
post: result = self.Buchungssatz->iterate
      (b:Buchungssatz; r:Bestand = 0 |
      if b.KontoNr    = n and b.Zeitpunkt <= z
                then  r + b.Menge
      endif
      )
```

Eine umfassende Beschreibung der Syntax und Semantik von OCL findet sich in der Spezifikation von OCL[426].

3.5.2.9 TemporalOCL – Temporal Object Constraint Language

Unter Temporal Object Constraint Language[427] (TemporalOCL) wird eine formale Notation zur Beschreibung von Reihenfolgebeziehungen der Verwendung von Diens-

[426] Vgl. OMG 2003, Kapitel 2.

ten und Synchronisationserfordernissen zwischen Diensten verstanden, welche als Erweiterungen von OCL entwickelt wurden.

Bei TemporalOCL kann zwischen vergangenheits- und zukunftsgerichteten Operatoren unterschieden werden, welche zur Beschreibung der Koordination zwischen Diensten dienen. Auf zeitpunktbezogene Operatoren wird verzichtet, da diese über das Eintreten eines bestimmten Zustandes abgebildet werden können. Im Folgenden werden die verschiedenen Operatoren in Kurzform erläutert.

Vergangenheitsgerichtete Operatoren

- *sometime_past φ*
  Ausgehend vom aktuellen Zustand hat irgendwann in der Vergangenheit einmal φ gegolten.

- *always_past φ*
  Ausgehend vom aktuellen Zustand hat in der Vergangenheit immer φ gegolten.

- *φ always_since_last ψ*
  Beginnend mit dem Zustand, in dem zuletzt ψ galt, hat seitdem bis zum aktuellen Zustand immer φ gegolten.

- *φ sometime_since_last ψ*
  Beginnend mit dem Zustand in dem zuletzt ψ galt, gab es vor dem aktuellen Zustand mindestens einen Zustand, in dem φ galt.

Zukunftsgerichtete Operatoren

- *sometime φ*
  Ausgehend vom aktuellen Zustand wird in mindestens einem zukünftigen Zustand φ gelten.

- *always φ*
  Ausgehend vom aktuellen Zustand wird in allen zukünftigen Zuständen φ gelten.

[427] Vgl. Ramakrishnan; McGregor 1999.

- *φ until ψ*
  Ausgehend vom aktuellen Zustand gilt in allen zukünftigen Zuständen φ, solange Zustand ψ erfüllt ist.

- *φ before ψ*
  Ausgehend vom aktuellen Zustand wird in einem zukünftigen Zustand einmal φ gelten, bevor ein Zustand erreicht wird, in dem ψ erfüllt ist.

Sonstige temporale Operatoren

- *initially φ*
  Im Anfangszustand (bei Erzeugung des Betrachtungsobjektes) gilt φ.

Weitere Operatoren

- *before(dienst1(para1,para2))*
  Dieser Ausdruck ist genau zu dem Zeitpunkt wahr, zu dem der *dienst1* mit den Parametern *para1* und *para2* aufgerufen wird.

- *after(dienst1(para1,para2))*
  Dieser Ausdruck ist genau zu dem Zeitpunkt wahr, in welchem die Abarbeitung von *dienst1*, der mit den Parametern *para1* und *para2* aufgerufen wurde, gerade erfolgreich beendet ist.

Im Folgenden wird ein Beispiel für eine TemporalOCL-Datei aufgeführt:

```
Auftragsabwicklung::RechnungDrucken(at:Auftrag)
 pre: sometime_past( after( KundenauftragAnnehmen(at) ) ) and
      not( after( AuftragStornieren(at) )
      sometime_since_last( after( KundenauftragAnnehmen(at) ) )

Auftragsabwicklung::KundenauftragAnnehmen(at:Auftrag)
 post: sometime( after( RechnungDrucken(at) ) ) or
       sometime( after( AuftragStornieren(at) ) )
```

In den letzten Jahren gab es verschiedene unabhängige Vorschläge von Sprachen zur Spezifikation temporaler Beziehungen, u. a. von Turowski[428] und Flake[429]. In dieser Arbeit wurde die TemporalOCL-Variante von Turowski gewählt.

[428] Vgl. Turowski; Conrad 2001.
[429] Vgl. Flake2003.

3.5.2.10 Semantisch normierte Orthosprachen, Ontologien und Terminologien

Zur Modellierung von Anwendungsarchitekturen und als ein Mittel zur Integration eignen sich inhaltlich standardisierte Sprachen, sogenannte Domain Specific Languages[430]. Eine solche inhaltsstandardisierte Sprache wird auch Normsprache (Rationalsprache/Zwischensprache) genannt[431]. Der Begriff „Normsprache" geht auf die rekonstruierte „Orthosprache" (gr. orthos, richtig) von Lorenzen[432] zurück. Damit sollte eine Fehlinterpretation der Normsprache (Orthosprache) als die einzig richtige Sprache, etwa im Sinne der Idealsprache, vermieden werden. Eine Normsprache ist eine geklärte Umgangssprache, die durch methodische Rekonstruktion einer in einem Anwendungsbereich eingesetzten natürlichen Sprache (Gemeinsprache/Fachsprache) gewonnen wird, z. B. indem vage und homonyme Benennungen entfernt und synonyme Benennungen nur in kontrollierter Form zugelassen werden[433]. Sie ist keine Sprache, die allumfassend durch Normierungsgremien starr genormt ist, sondern vielmehr eine in einer Sprachgemeinschaft nach Regeln dynamisch genormte (Fach-)Sprache.

Das Besondere an einer Normsprache im Vergleich zu formalen Sprachen ist ihr materialsprachlicher Charakter bzw. ihr inhaltlicher Bezug auf einen fachlichen oder praktischen Gegenstandsbereich. Eine Normsprache wird im Hinblick auf ihr konkretes Anwendungsgebiet nicht nur in ihrer Grammatik (formaler Teil), sondern auch in ihrem Bestand zulässiger Themenwörter (materialer Teil) – der einem bestimmten Anwendungsgebiet zugeordnet ist – zur Bildung korrekter Aussagen über dieses Gebiet vereinbart. Zur Festlegung des Wortschatzes wird ein entsprechendes Wörterbuch auf der Basis einer Repositorium-Anwendung aufgebaut und gepflegt.

Die Wörter der Normsprache unterscheiden sich von Wörtern der natürlichen (empirischen) Sprache grundsätzlich darin, dass sie nicht erst in einem bestimmten Anwendungskontext eine bestimmte Bedeutung annehmen, sondern als Terminologie eines Gegenstandsbereichs, die heute einer domänenspezifischen Ontologie nahe kommt, für stets dieselbe Verwendung vorgesehen sind[434]. Dagegen beschränken sich formale Sprachen auf die Festlegung der Grammatik (z. B. Satzbaupläne) gültiger Aussagen eines Anwendungsgebiets. Ausdrücke in einer formalen Sprache sind deshalb allein

[430] Vgl. Ortner 2005a, S. 257.
[431] Vgl. Schienmann 1997.
[432] Vgl. Lorenzen 1973.
[433] Vgl. Lehmann 1998, S. 366.
[434] Vgl. Kamlah; Lorenzen 1996.

gültig aufgrund ihrer Form, unabhängig davon, welche Themenwörter bzw. Fachwörter sie enthalten.

Auf Basis einer solchen reglementierten Sprache mit einer normierten Terminologie und festgelegten Satzstrukturen, kann ein Anwendungsgebiet inhaltlich modelliert und klassifiziert werden. „Inhaltlich" kann hier auch synonym mit „semantisch" verwendet werden, während „formal" oder „strukturell" auch „syntaktisch" (z. B. Satzbaupläne) genannt wird. Die auf diese Art normierten Aussagen können leicht in Diagrammsprachen übersetzt werden[435]. Daher eignen sich Normsprachen auch als Zwischensprache, um den Übergang von umgangssprachlichen Aussagen zu formalisierten Aussagen, wie bspw. Baumstrukturen, Terminologien und Ontologien, effizient zu organisieren.

3.6 Anwendungssystemarchitekturen

Im Folgenden werden Anwendungssystemenarchitekturen vorgestellt, die in den letzten Jahren von der Anwendungsinformatik durch CORBA[436] inspiriert und diskutiert wurden.

3.6.1 E-NOgS³

Mit Electronic New Organon-{Service, Servant, Server} (E-NOgS³) liegt eine von Ortner[437] spezifizierte Anwendungssystemarchitektur für E-Commerce-Anwendungen vor, die allgemeine Dienste bereitstellt. Diese Plattform sichert die Integration von Anwendungen auch von Drittanbietern sowie die Zugriffsmöglichkeiten von Anwendern auf das System. Weiterhin wird Transaktionalität erreicht, d. h. die Überwachung und Steuerung von Aktionen der physischen Welt über das System. Auch die Unterstützung der Interaktion zur Schaffung eines gleichen Begriffsverständnisses ist Gegenstand dieser Plattform.

E-NOgS³ wurde aus den Gebrauchssprachen der Anwender durch Unterscheidung zwischen generischen (Middleware) und spezifischen (Anwendungssoftware) Sprachhandlungen rekonstruiert und in Basissysteme untergliedert[438]. Abbildung 41 stellt die Architektur von E-NOgS³ grafisch dar. Technologisch gesehen handelt es sich im Kern um ein Middleware-Framework, da die Dienste zwischen Betriebssystem und

[435] Vgl. Vogler 1994.
[436] Vgl. Schulze 1999.
[437] Vgl. Ortner; Overhage 2003a.
[438] Vgl. Ortner; Overhage 2003a, S. 24.

anwendungsspezifischen Teilen angeordnet sind. Durch Kopplung an diesen anwendungsspezifischen Teil kann das Middleware-Framework seine Funktionalität vollständig entfalten. Mit E-NOgS³ wurde das Ziel einer generischen Referenzarchitektur für komponentenbasierte Anwendungssysteme im E-Commerce-Umfeld verfolgt[439].

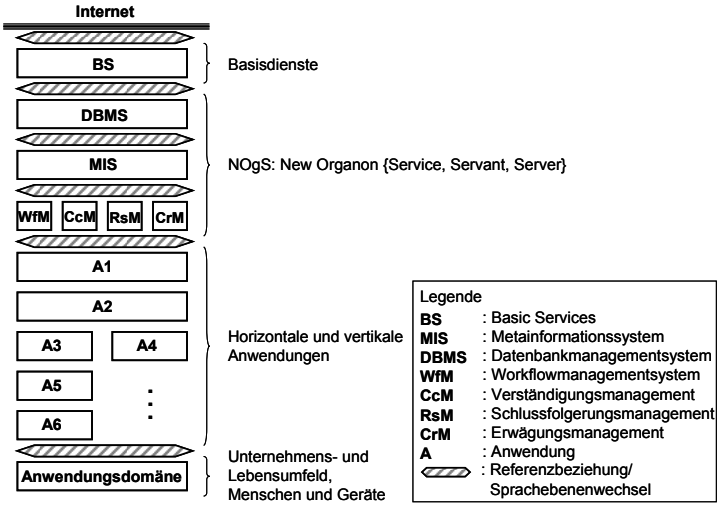

Abbildung 41: E-NOgS³-Referenzarchitektur[440]

Im Folgenden werden die wesentlichen Elemente von E-NOgS³ beschrieben.

Basic Services

Die Basisdienste sind für die Zusammenarbeit der Komponenten notwendig, denn in dieser Schicht ist die Kommunikation der Komponenten realisiert. Konkret handelt es sich um eine XML-Web Service-Kommunikationsstruktur. Diese Art der Kommunikation erlaubt es, aus einer Anwendung heraus eine Methode einer anderen Anwendung aufzurufen. Dies kann über ein Netzwerk geschehen oder lokal auf einem Server mit dem XML-Web Service ablaufen. Insofern entspricht der Service einem Remote Procedure Call[441] (RPC) also einem entfernten Prozeduraufruf. Mit Hilfe von Web Services können Agenten direkt mit einer solchen Anwendung interagieren, ohne über die Benutzungsschnittstelle für Anwender (umständlich) zugreifen zu müssen. Den Anwendungen und Komponenten innerhalb des Frameworks steht eine bidirektionale

[439] Vgl. Ortner; Overhage 2003a, S. 22.
[440] In Anlehnung an Ortner; Overhage 2003a, S. 24.
[441] IETF; Srinivasan 1995.

Kommunikation über diesen Service oder einen direkten Aufruf zur Verfügung. Die Integration und der Zugriff wird durch diese Implementierung der Basisdienste verbessert.

Datenbank-Managementsystem

Das Datenbank-Managementsystem besteht aus einer Datenbank mit kontrollierender Verwaltung der Datenhaltung. Ziel ist die effiziente, aber dennoch sichere Datenhaltung. Dazu gehört die Sicherstellung allgemeiner Prinzipien der Datenhaltung wie die Unterstützung der Transaktionalität, aber auch die Gewährleistung von Sicherheitsstandards bezogen auf den Zugriff der Daten.

Metainformationssystem (Repositorium)

Um die installierte Software, die erzeugten Daten und abgebildeten Strukturen (modellierte Geschäftsprozesse etc.) verwalten zu können, muss ein Metainformationssystem zur Verfügung stehen. Dieses Repositorium dient als Dokumentationssystem[442]. Es kann dabei beliebig strukturierte Informationen von Programmen bis zu Textdokumenten und Schemata speichern. Entsprechend greifen neben technischen Systemen wie Datenbank-Managementsystemen auch Anwender und Entwickler darauf zu. Je nach Zweck des E-NOgS³-Systems kommen Entwicklungsrepositorien zum Einsatz, die Montagepläne und Verwendungsnachweise verwalten können, oder Konzepte wie Data-Dictionaries und Kataloge, die Metainformationen speichern[443].

Workflow-Management-System

Das Workflow-Management-System unterstützt die Interaktion von E-Commerce-Systemen, indem es die Prozesse der Handelsphasen in geeigneter Weise abbildet[444]. Dabei dient ein Vorgangssteuerungssytem als Komponente, die als primären Gegenstand das Geschehen in einer Organisation modelliert und die Anwender elektronisch bei der Abarbeitung der Geschäfts-, Informations- und Kontrollprozesse unterstützt[445].

Schlussfolgerungs-, Verständigungs- und Erwägungssystem

Diese Dienste dienen der Verbesserung der Kommunikation und Interaktion der beteiligten Kommunikationspartner. Das Verständigungssystem, auch Kommunikations-Management-System genannt, fördert die sprachliche Ausdrucksweise der Kommuni-

[442] Vgl. Ortner 1999, S. 235f.
[443] Vgl. Ortner 1999, S. 237.
[444] Vgl. Ortner; Overhage 2003a, S. 25.
[445] Vgl. Jablonski 1997, S. 23.

kationspartner und dient zur Überwindung von Heterogenitäten der Kommunikationspartner insbesondere semantischer Art. Erwägungs- und Schlussfolgerungsdienste (z. B. Expertensysteme) unterstützen den Aspekt der Kreativität des Systems[446]. Allgemein dienen diese Dienste zur Unterstützung der Mensch-Maschine-Interaktion. Ein sogenanntes Reasoning-Management-System umfasst Schlussfolgerungs-, Verständigungs- und Erwägungssystem.

3.6.2 Service-orientierte Architektur (SOA)

Eine Service-orientierte Architektur[447] (SOA, engl. Service-oriented Architecture; auch diensteorientierte Architektur genannt) ist ein Konzept für eine themengebietorientierte Systemarchitektur, in dem Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und fachlich lose gekoppelten Services implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Serviceinteraktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt. Mit einer Service-orientierten Architektur werden in der Regel die Gestaltungsziele der Geschäftsprozessorientierung, der Wandlungsfähigkeit (Flexibilität), der Wiederverwendbarkeit und der Unterstützung verteilter Softwaresysteme verbunden. Somit ist eine SOA primär ein Managementkonzept und setzt erst in zweiter Linie ein Systemarchitekturkonzept voraus[448]. Das Managementkonzept strebt eine an den gewünschten Geschäftsprozessen ausgerichtete IT-Infrastruktur an, die schnell auf veränderte Anforderungen im Geschäftsumfeld reagieren kann. Das Systemarchitekturkonzept sieht die Bereitstellung fachlicher Dienste und Funktionalitäten in Form von Services vor. Dieses Systemarchitekturkonzept muss in konkreten Anwendungssituationen (in Unternehmen) entwickelt werden. Daraus resultiert dann eine Unternehmens-Anwendungs-Architektur auf SOA-Basis (siehe Abbildung 42).

Geschäftsprozesse lassen sich durch Aneinanderreihung von Serviceaufrufen (Komposition von Services) realisieren bzw. unterstützen. Die Programmlogik ist nicht in einem einzigen Programm zu finden, sondern verteilt über mehrere unabhängige Dienste[449]. Damit lässt sich eine SOA als Blueprint eines komponenten- bzw. prozessorientierten Anwendungssystems auffassen.

---

[446] Vgl. Ortner; Overhage 2003a, S. 26.
[447] Vgl. Dostal et al. 2005.
[448] Vgl. Brabänder; Klückmann 2006, S. 32.
[449] Vgl. Krafzig; Banke; Slama 2005, S. 56.

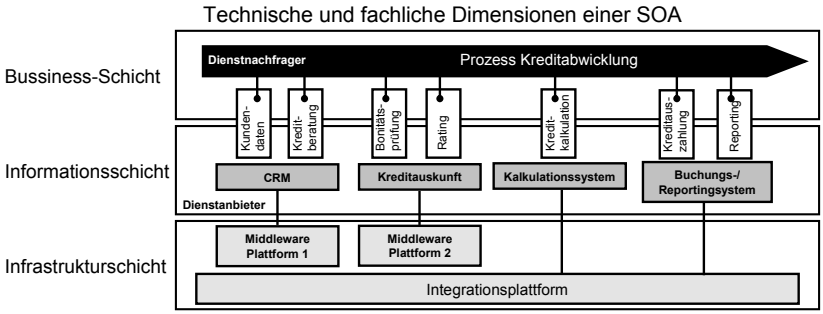

Abbildung 42: SOA-Schichtenarchitektur[450]

Ein Service ist in diesem Kontext als eine (betriebswirtschaftliche) Funktionalität definiert, die über eine standardisierte Schnittstelle in Anspruch genommen werden kann. Er ist damit eine spezielle Implementierung einer (Software-)Komponente. Die SOA sieht eine Menge voneinander unabhängiger, lose gekoppelter Dienste (Services) vor. Ein Dienst wird von einem Dienstanbieter (engl. service provider) angeboten. Ein Dienstnachfrager (engl. service consumer) stellt eine Anfrage (engl. service request) an einen Dienst und bekommt daraufhin eine Antwort (engl. service response) vom Anbieter.

Häufig werden für SOAs Web Services auf Basis der wenigen bestehenden Standards wie SOAP, WSDL und UDDI eingesetzt, doch kann eine SOA prinzipiell auf jeder dienstbasierten Technologie wie zum Beispiel CORBA, DCOM oder Enterprise Java Beans (EJB) aufgebaut werden. Da Services in unterschiedlichen Programmiersprachen und auf unterschiedlichen Systemplattformen realisiert werden können, wird eine SOA häufig zur Anwendungsintegration genutzt.

Ein weiteres wesentliches Ziel einer SOA ist die Kapselung von persistenten Daten durch Dienste, die exklusives Lese- und Schreibrecht auf „ihre“ Daten besitzen. Die hierdurch erzielte Modularität führt zu geringen Redundanzen und einer höheren Flexibilität der Anwendungssysteme, was häufig zu niedrigeren Betriebskosten führt. Allerdings entsteht zusätzlicher Aufwand bei der Migration durch Entwicklung geeigneter Adapter, um Geschäftslogik (engl. business logic) und (betriebswirtschaftliche) Funktionalität bereits existierender Anwendungen als Services anzubieten[451].

---

[450] In Anlehnung an Niemann 2006, S. 6.
[451] Vgl. Patrick 2005, S. 845.

4 Externes Anwendungsmanagement auf der Basis von (Software-)Komponenten und einem (Software-)Komponenten-Handel für mobile Anwendungen

4.1 Integration des externen Anwendungsmanagements in mobile verteilte Systeme

In diesem Abschnitt wird das Anwendungsmanagement (siehe Abschnitt 2.5) in verteilte Systeme integriert. Dafür wird modellbasiert vorgegangen und die Prinzipien des Anwendungsmanagements in Modelle verteilter Systeme eingearbeitet, um daran die besonderen Eigenschaften herzuleiten und zu untersuchen. Hierbei wird internes Anwendungsmanagement (lokale Verwaltung) und externes Anwendungsmanagement (Integration externer Dienste) unterschieden (siehe Abschnitt 2.5.2).

Das interne Anwendungsmanagement kümmert sich um die lokale Verwaltung der auf einem Endgerät existierenden Anwendungen bzw. Komponenten. Sobald jedoch das Management der Kommunikation zwischen Anwendungen auf verschiedenen, z. B. auch externen Einheiten beachtet wird, kommt der Aspekt der Integration weiterer Komponenten zum Tragen. Komponenten müssen ggf. von außen bereitgestellt werden, dies ist Aufgabe des externen Anwendungsmanagements.

Am Anfang der Entwicklung von verteilten Systemen standen unabhängige Einheiten, die miteinander in Kontakt treten sollten. Um einen Datenaustausch zu ermöglichen, musste eine einheitliche Sprache (Normsprache bspw. in Form von Standards bzw. Protokollen) zwischen diesen Einheiten definiert werden, so dass eine Verständigung erst ermöglicht wurde. Diese Normsprache musste von allen Einheiten, die an dieser Kommunikation teilnehmen, verstanden werden. Um nun aber auch im Betrieb die Möglichkeit des Datenaustauschs herzustellen, mussten sich auch die einzelnen Einheiten miteinander verständigen können.

In Abbildung 43 treten die Einheiten in Kontakt durch eine Kommunikationsschicht, die absichtlich nicht näher spezifiziert ist, um den allgemeinen Charakter der Betrachtung zu wahren. Diese Schicht kann real existieren oder auch nur rein virtuellen Charakter haben. Der in dieser Schicht auftretende Austausch geschieht in der von allen Einheiten verstandenen Normsprache. Um nun die Bedeutung des Anwendungsmana-

gements unter dem Aspekt der Kommunikation einzuleiten, ist zu überlegen, wie das Anwendungsmanagement in einem solchen modellhaften Kommunikationssystem platziert werden kann.

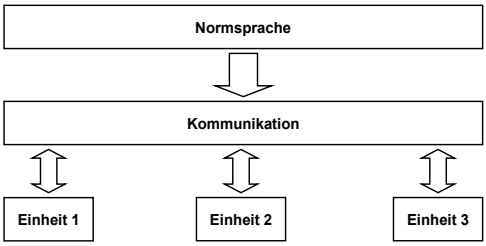

Abbildung 43: Grundstruktur eines verteiltes Systems

Um Interaktion zwischen verschiedenen Anwendungen, Anwendungsteilen oder Komponenten verwalten zu können, muss das Anwendungsmanagement die Normsprache verstehen und anwenden können. Es muss beobachten können, welche Tätigkeiten Anwendungen ausführen und sie durch bestimmte Befehle beeinflussen können. Durch Berücksichtigung der Interaktion muss das Anwendungsmanagement direkten Einfluss auf die Kommunikation nehmen können. Das Anwendungsmanagement fügt sich daher in das Modell eines verteilten Systems gemäß Abbildung 44 ein. Die Beziehungen zwischen Anwendungsmanagement und der Kommunikationsschicht bzw. den Einheiten sind hierbei allgemeiner Natur und werden deshalb nicht näher spezifiziert.

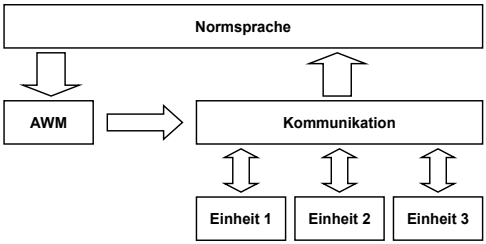

Abbildung 44: Allgemeine Struktur eines verteilten Systems mit integriertem Anwendungsmanagement

Basierend auf den in Abschnitt 3.1.3.4 beschriebenen Möglichkeiten der Aufbauorganisation von verteilten Systemen in klassische, ad-hoc und nomadische Netzwerke lassen sich auch drei verschiedene Anordnungen des Anwendungsmanagements in Bezug auf die Kommunikation und deren beteiligten Einheiten modellieren.

4.1.1 Anwendungsmanagement in klassischen Netzwerken

Für das klassische Netzwerk regelt das Anwendungsmanagement die gesamte Kommunikation. Da die Kommunikation absolut statisch ist und zusätzlich im Falle eines klassischen Netzwerks auch real (z. B. in Form von Netzwerkknoten und Hubs) existiert, kann das Anwendungsmanagement die Kommunikation vollständig kontrollieren, indem es die zugehörige Schicht kontrolliert. Aufgrund dieser dominanten Rolle wird das Anwendungsmanagement in einem klassischen Netzwerk als Rahmen um die Kommunikation modelliert (siehe Abbildung 45).

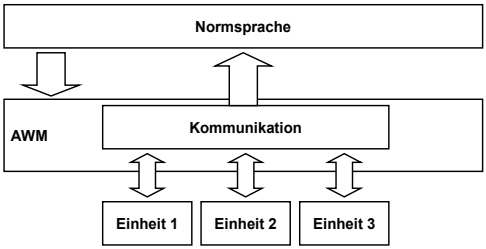

Abbildung 45: Anwendungsmanagement in einem klassischen Netzwerk

Die Betrachtung der klassischen Netzwerke dient hier der Darstellung der Unterschiede des Anwendungsmanagements in den verschiedenen verteilten Systemen.

4.1.2 Anwendungsmanagement in ad-hoc Netzwerken

Im ad-hoc Netzwerk kann das Anwendungsmanagement keine dominante Rolle einnehmen. Da ad-hoc Netzwerke direkte Verbindungen zwischen den einzelnen Einheiten und somit den einzelnen Anwendungen nutzen und es nur eine virtuelle Schicht der Kommunikation zwischen diesen Einheiten gibt, kann das Anwendungsmanagement die Kommunikation nicht derart kontrollieren, dass es die Kommunikationsebene beeinflusst. Ganz im Gegenteil, das Anwendungsmanagement kann nur realisiert werden, indem es die Kommunikation zwischen den Einheiten durch Kontrolle der Einheiten selbst erreicht. Das gesamte Anwendungsmanagement muss verteilt auf den einzelnen kommunizierenden Einheiten vorliegen und von dort aus die Kontrolle über die aus- und eingehende Kommunikation ausüben. Abbildung 46 zeigt ein entsprechendes Modell, wobei die Pfeile zwischen Normsprache und den externen Einheiten aus Übersichtsgründen weggelassen wurden.

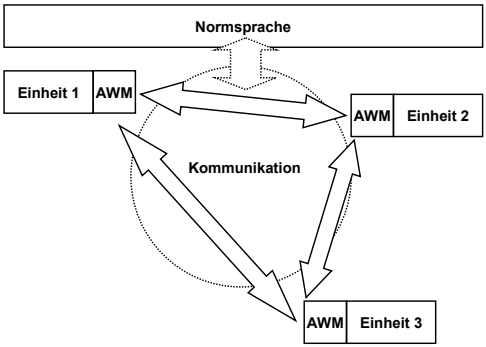

Abbildung 46: Anwendungsmanagement in einem ad-hoc Netzwerk

Aufgrund der Eigenschaften dieser Netzwerke (siehe Abschnitt 3.1.3.4) ergeben sich Anforderungen in Bezug auf wechselnde Bandbreiten der Verbindung bis hin zu plötzlichen Verbindungsabbrüchen. Im Gegensatz zum klassischen Netzwerk, bei dem die Verbindungen allgemein als statisch und fest angesehen werden können, muss hier besonders darauf geachtet werden, trotz dynamischer Verbindungen ein effizientes Anwendungsmanagement zu ermöglichen.

Ein weiterer, für das Anwendungsmanagement in ad-hoc Netzwerken, wichtiger Aspekt ist die Skalierbarkeit. Ein Anwendungsmanagement sollte ein solches Netzwerk möglichst unabhängig von der Netzwerkgröße verwalten können. Speziell in ad-hoc Netzwerken kann jederzeit eine große Zahl von neuen Endgeräten das Netz verlassen oder sich damit verbinden, so dass schon allein aufgrund dieser Dynamik der Aspekt der Skalierbarkeit besonders hervortritt.

Ein zusätzliches Problem und damit auch eine Anforderung an das Anwendungsmanagement ist die Gewährleistung von Sicherheit der Verbindungen. In einem stationären Netzwerk konnte das Anwendungsmanagement schon alleine durch seine steuernde Funktion im Netzwerk einen hohen Sicherheitsstandart ohne großen zusätzlichen Aufwand garantieren. In einer derart dynamischen Umgebung ist dies nicht mehr in dieser Art möglich. Daher muss das Anwendungsmanagement nun sehr viel stärker zur Sicherheit beitragen.

Ein wichtiger Aspekt ist die bereits beschriebene starke Heterogenität der möglichen Endgeräte in einem solchen Netzwerk, so dass das Anwendungsmanagement plattformübergreifend arbeiten muss. Gerade auch in ad-hoc Netzwerken spielt der Res-

sourcenverbrauch durch das Anwendungsmanagement eine große Rolle. In einem Umfeld, das von knappen Ressourcen geprägt ist, kann kein aufwändiges Anwendungsmanagement auf den mobilen Einheiten eingesetzt werden, das die zur Verfügung gestellten Ressourcen zu stark beansprucht.

4.1.3 Anwendungsmanagement in nomadischen Netzwerken

Nomadische Netzwerke weisen eine real existierende Kommunikationsschicht auf. Somit ist es möglich, die Kommunikation an dieser Schicht selbst zu kontrollieren und das Anwendungsmanagement in das Kommunikationssystem zu integrieren. Das Anwendungsmanagement muss jedoch den Kommunikationsaspekt auch auf den einzelnen Einheiten selbst beachten, da diese evtl. von der Kommunikationsbasis getrennt werden können und dennoch funktionsfähig bleiben müssen. Man könnte daher nomadische Netzwerke wie ad-hoc Netzwerke behandeln, und auf die Möglichkeit das Anwendungsmanagement auch in der Kommunikationsschicht zu verankern verzichten. Aufgrund der begrenzten Ressourcen der einzelnen, speziell der mobilen, Einheiten ist es sinnvoll möglichst viel Rechenaufwand und Speicherplatz von ihnen fernzuhalten. Daher wäre dieser Verzicht von Nachteil.

Da die Kommunikationsschicht als reale Schicht mit sehr viel Rechenkapazität ausgestattet werden kann, die einzelnen Einheiten aber nicht, sollte das Anwendungsmanagement so weit wie möglich in die Kommunikationsschicht verlagert werden. Nur das für die evtl. Trennung vom Netzwerk Nötige sollte in den mobilen Einheiten selbst belassen werden. In einem solchen Netzwerk liegt eine starke Verteilung der Ressourcen vor. Damit ergeben sich völlig neue Möglichkeiten in Bezug auf eine Lastverteilung zwischen verschiedenen Einheiten, die ein Anwendungsmanagement aufgrund des hohen Potenzials nutzen sollte.

Die bei ad-hoc Netzwerken beschriebenen Aspekte der dynamischen Verbindungen, der Abbruchsicherheit, der Skalierbarkeit, der Plattformunabhängigkeit, wie auch der Sicherheit gelten hier genauso, können jedoch zusätzlich kombiniert werden mit den bereits ausgereiften Technologien der klassischen Netzwerke. Hinzu kommt jedoch der Aspekt der Client/Server-Aufteilung. Hierbei können aufgrund der unterschiedlichen Ressourcen der Geräte verschiedenste, auch dynamische Möglichkeiten einer Client/Server-Struktur entstehen. Ein weiterer Aspekt besteht in der Verwaltung und Vermittlung von Diensten in einem nomadischen Netzwerk. Dies kann durch eine

zentrale Dienstvermittlung, aber auch dezentral oder hybrid (zentral und dezentral) erfolgen (siehe Abbildung 47).

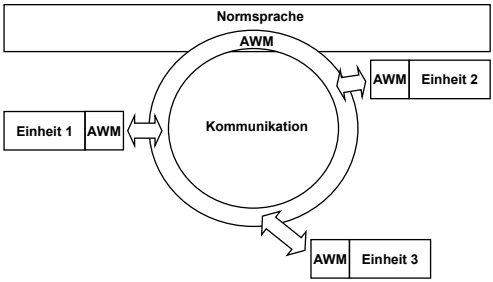

Abbildung 47: Anwendungsmanagement in einem nomadischen Netzwerk

Insgesamt entsteht damit ein Anwendungsmanagement, das zu großen Teilen auf der Kommunikationsschicht zu finden ist, aber auch in den einzelnen Einheiten existieren muss. Dieses Modell eignet sich insbesondere für einen mobilen Marktplatz, dessen Marktplatzsystem an eine stationäre Infrastruktur angeschlossen ist und bei dem der Zugang der mobilen Endgeräte über unzuverlässige (mobile) Netzwerke realisiert ist.

4.2 (Software-)Komponenten und (Software-)Komponenten-Handel

Nun erfolgt eine Einschränkung des externen Anwendungsmanagements für den Fall komponentenbasierter Anwendungssysteme. Dabei wird zunächst der Begriff „(Software-)Komponente“ vertiefend diskutiert und dazu korrespondierende Komponentenmodelle vorgestellt sowie die Möglichkeiten des (Software-)Komponenten-Handels beschrieben.

4.2.1 (Software-)Komponenten

Der Begriff „(Software-)Komponente“ wird nicht einheitlich verwendet. Daher erfolgt zunächst ein Überblick gängiger Definitionen. Aus diesen Definitionen werden die wesentlichen Merkmale von (Software-)Komponenten abgeleitet. Ein Vorschlag des GI-Arbeitskreises „Komponentenorientierte betriebliche Anwendungssysteme (AK 5.10.3)“ zur Vertiefung des Komponentenbegriffs mit Hilfe einer Einteilung von Komponentenarten wird aufgegriffen und das Paradigma der komponentenorientierten Softwareentwicklung vorgestellt.

4.2.1.1 Definitionen

Eine vielfach zitierte Definition von (Software-)Komponenten ist das Ergebnis des *„Workshop on Component-Oriented Programming“* im Rahmen der zehnten europäischen Konferenz über Objektorientiertes Programmieren (ECOOP '96): *„A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.*[452]*“* Diese Definition wird in vielen Veröffentlichungen allein Szyperski zugerechnet. Szyperski weist in der zweiten Auflage seines Standardwerks „Component Software“[453] aber darauf hin, dass diese Definition im Rahmen des Workshops und nicht von ihm allein entwickelt wurde.

Nach dieser Definition zeichnet sich eine Komponente durch drei charakteristische Eigenschaften aus[454]:

- eine Komponente ist eine unabhängig einsetzbare Einheit,
- eine Komponente wird von Dritten mit anderen Komponenten zu einem Kompositum verbunden, und
- der Zustand einer Komponente ist nicht persistent.

Um als unabhängig einsetzbare Einheit gelten zu können, muss eine Komponente von ihrer Umgebung, insbesondere anderen Komponenten, klar abgegrenzt werden. Sie kapselt ihre Funktionalität und stellt diese nur über vertraglich festgelegte, definierte Schnittstellen nach außen bereit. Dabei sind die Softwarekontrakte meist als Vor- und Nachbedingungen einer über eine Schnittstelle angebotenen Funktion oder als deren Invariante zu verstehen. Wird die Komponente in einen Kontext gestellt, der ihren Vor- und Nachbedingungen genügt, so erfüllt sie ihren Teil des Vertrages. Dritte – dies sind in der Regel nicht die Hersteller der Komponenten, sie haben somit keinen Zugriff auf Implementierungsdetails der Komponente – fügen verschiedene Komponenten zu einem Kompositum, einer Anwendung, zusammen[455].

In dieser von Szyperski vertretenen Definition bleibt die Nicht-Persitenz der Zustände von Komponenten kritisch anzumerken. Denn durch Hinzunahme von bspw. Datenbank-Konnektoren haben einzelne Komponenten die Möglichkeit der persistenten Da-

[452] Szyperski; Pfister 1997, S. 130.
[453] Szyperski; Gruntz; Murer 2002, S. 41.
[454] Vgl. Szyperski 1998, S. 36.
[455] Vgl. Szyperski 1998, S. 34.

tenhaltung. Ebenfalls kommt es auf die Granularität der Komponente an. Mehrere zusammengesetzte Komponenten mit organisierter Datenhaltung sind selbst auch wieder eine Komponente groberer Granularität mit dem Zustand der Persistenz.

Eicker und Nietsch verwenden das Wort „Components" und definieren damit den Begriff „Komponente" wie folgt: *„Components repräsentieren speziell im Hinblick auf Wiederverwendung entworfene und implementierte Softwarebausteine. Ein Baustein stellt eine Menge von öffentlichen Diensten (engl. public services) zur Nutzung bereit.* [456]*"*

Ackermann et al.[457] verstehen den Begriff der Komponente in einer weiter gefassten Definition. Eine Komponente besteht aus verschiedenartigen (Software-)Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten eingesetzt werden, die zur Zeit der Entwicklung nicht vorhersehbar sind. Hierbei werden alle Artefakte, die einer Komponente zuzuschreiben sind, eingeschlossen, also neben der Komponente im Binärformat auch ihre Dokumentation, in ihr verankerte Grafiken, Voreinstellungen oder Testfälle. Zudem wird mit der Eigenschaft der Vermarktbarkeit verlangt, dass die Komponente als eigenständiges wirtschaftliches Gut bspw. auf offenen aber auch unternehmensinternen Marktplätzen gehandelt werden kann.

Eine sehr ausführliche Diskussion der Frage, was eine (Software-)Komponente charakterisiert, führten Broy, Deimel, Henn, Koskimies, Plášil, Pomberger, Pree, Stal und Szyperski per E-Mail, die in der Zeitschrift „Software – Concepts and Tools"[458] veröffentlicht wurde. Im Hinblick auf die Verwendung in der Unified Modelling Language (UML) diskutieren Stevens und Pooley[459] den Begriff „Komponente".

Ein wesentliches Konzept von Komponenten ist die Bereitstellung von Diensten, die durch andere genutzt werden können, wobei die Nutzer selbst Komponenten oder andere Arten von Software sein können.

---

[456] Eicker; Nietsch 1999, S. 363.
[457] Vgl. Ackermann et al. 2002, S. 1.
[458] Broy et al. 1998.
[459] Stevens; Pooley 2000, S. 253ff.

4.2.1.2 Komponentenarten

Ackermann et al.[460] verfeinern die Unterscheidung von Komponenten in generische und spezifische Komponenten durch die Einführung des Begriffs „Fachkomponente“. Eine Fachkomponente ist eine Komponente, die eine bestimmte Menge von Diensten eine Anwendungsbereichs (Domäne) anbietet.

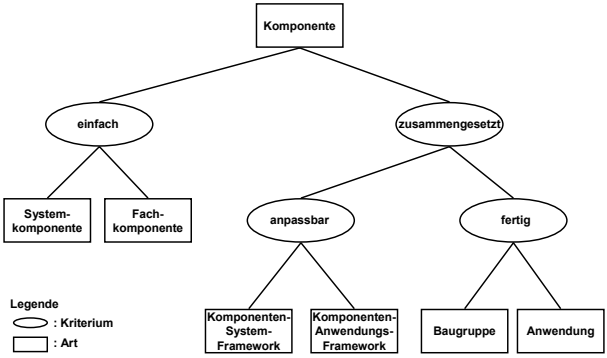

Abbildung 48: Einteilung von Komponentenarten[461]

Darüber hinaus können noch weitere Komponentenarten unterschieden werden, wie in Abbildung 48 dargestellt[462]. Komponenten lassen sich in einfache, aber dennoch abgeschlossene, und zusammengesetzte Komponenten einteilen. Neben den bereits erwähnten Fachkomponenten zählen die Systemkomponenten, die generische Funktionen realisieren, zu den einfachen Komponenten. Die zusammengesetzten Komponenten lassen sich weiter in anpassbare und fertige unterteilen. Fertig zusammengesetzte Komponenten sind dann zum einen Baugruppen, die aus System- und bzw. oder Fachkomponenten zusammengesetzt sind (diese werden auch Konfigurationen genannt) und zum anderen Anwendungen, die als große, unabhängig einsetzbare Komponenten aufzufassen sind, jedoch auch noch mit anderen Komponenten verbunden werden können. Ein Komponenten-System-Framework erfüllt generische, ein Komponenten-Anwendungs-Framework anwendungsspezifische Funktionen. Beide lassen sich als zusammengesetzte Komponenten anpassen, indem sie bspw. um weitere einfache oder auch zusammengesetzte Komponenten zu einer fertigen Anwendung erweitert werden.

---

[460] Vgl. Ackermann et al. 2002, S. 1.
[461] In Anlehnung an Ackermann et al. 2002, S. 2.
[462] Vgl. Ackermann et al. 2002, S. 2.

4.2.1.3 Wiederverwendung von Komponenten

Orthogonal zu der Einteilung von Komponentenarten kann nach der Art der Wiederverwendung von Komponenten unterschieden werden. Im klassischen Sinne werden dabei Black-Box- und White-Box-Ansätze verstanden. Später entwickelten sich mit Grey-Box und Glass-Box Hybridformen der klassischen Ansätze.

Black-Box

Im Falle einer Black-Box-Wiederverwendung von Komponenten geht man davon aus, dass die Komponente vollständig gekapselt ist, ihre innere Struktur und jegliche Implementierung nach außen verborgen bleibt. Damit wird eine maximale Unabhängigkeit von der Umgebung erzielt. Bekannt sind die Komponentenbeschreibung, ihre Funktionalität (wie sie beschrieben ist, also ohne Nachweis) und ihre Schnittstellen. Im Extremfall handelt es sich bei Black-Box-Komponenten um Komponenten, die ausschließlich in Binärcode vorliegen[463].

White-Box

Im Falle einer White-Box-Wiederverwendung von Komponenten geht man davon aus, dass die Komponente vollständig einsehbar und manipulierbar ist. Damit kann auch auf das Konzept der Vererbung zurückgegriffen werden. Bekannt sind die Komponentenbeschreibung, ihre Funktionalität, ihre Struktur, ihr Inhalt (sogar der Code) und die Schnittstellen[464].

Grey-Box

Im Falle der Grey-Box-Wiederverwendung wird eine teilweise Kapselung und bzw. oder Anpassung ermöglicht. Grey-Box vereinigt somit die Konzepte Black-Box und White-Box. Dabei treten verschiedene „Grauabstufungen“ auf. Die sichtbaren Teile einer Grey-Box-Komponente sind modifizierbar. Die gekapselten Teile einer Grey-Box-Komponente können weder eingesehen noch verändert werden. Unabhängig vom Grad der Kapselung dürfen keine versteckten Abhängigkeiten zwischen der Implementierung der Grey-Box-Komponente und ihrer Umgebung entstehen[465].

---

[463] Vgl. Griffel 1998, S. 418 und 19.
[464] Vgl. Griffel 1998, S. 419 und 18.
[465] Vgl. Griffel 1998, S. 419.

Glass-Box

Im Falle der Glass-Box-Wiederverwendung werden die Vorteile der sichtbaren Struktur der White-Box-Wiederverwendung mit der Nichtanpassbarkeit der Black-Box-Wiederverwendung kombiniert. Bekannt sind die Komponentenbeschreibung, ihre Funktionalität, ihre Struktur, ihr Inhalt (sogar Code) und die Schnittstelle. Anpassung ist nicht erlaubt. Unter den Aspekten der Dokumentation und der Vertrauenswürdigkeit (Qualitätsaspekt) unter Aufrechterhaltung der strikten Kapselung ist Glass-Box-Wiederverwendung sinnvoll einsetzbar[466].

Commercial Off-The-Shelf (COTS)-Komponente

Als COTS (engl. für kommerzielle Produkte aus dem Regal) werden seriengefertigte Komponenten bezeichnet, die in großer Stückzahl völlig gleichartig aufgebaut verkauft werden. Üblicherweise handelt es sich dabei um Software-Produkte, wie bspw. Büroanwendungen oder Warenwirtschaftssysteme. Dadurch, dass ab Werk keine Anpassungen an die Bedürfnisse des individuellen Kunden vorgenommen werden, erhofft sich der Nutzer weitgehende Kosteneinsparungen, da hier die Entwicklungskosten nicht vom Auftraggeber alleine, sondern vom Markt getragen werden. COTS lassen sich der Black-Box-Wiederverwendung zuordnen. Ein Unterscheidungsmerkmal von COTS und Black-Box ist die Granularität der vermarkteten Komponente, die im COTS-Fall grober ausfällt, da es sich eher um ganze Anwendungen handelt[467].

4.2.1.4 Paradigma der komponentenorientierten Anwendungsentwicklung

In der Literatur finden sich verschiedene Ansichten darüber, was komponentenorientierte Anwendungsentwicklung ist und welches Paradigma ihr zugrunde liegt.

Im Sinne von COTS-Wiederverwendung steht eher der Standardisierungsgedanke[468] von Software-Systemen bzw. Software-Produkten im Vordergrund.

Griffel führt die Komponentenorientierung auf das Prinzip der Wiederverwendung zurück[469]. Im Rahmen der Wiederverwendung kann eine Definition in Anlehnung an Biggerstaff und Perlis wie folgt verstanden werden: *„Wiederverwendung von Software ist das erneute Anwenden von bei der Entwicklung eines Softwaresystems entstandenen Artefakten und angesammeltem Wissen bei der Entwicklung eines neuen Software-*

[466] Vgl. Griffel 1998, S. 419.
[467] Vgl. Voas 1998.
[468] Vgl. Buxmann 1996.
[469] Vgl. Griffel 1998, S. 10.

systems, um den Aufwand zur Erstellung und Pflege dieses neuen Systems zu reduzieren.[470]*“*

Diese Definition geht in die Richtung der im IEEE Standard 610.12-1990[471] festgelegten Definition des englischsprachigen Begriffs „Reusability“ als Grad, zu dem ein Software-Modul oder ein anderes Entwicklungsdokument in mehr als einem Computer-Programm oder Softwaresystem verwendet werden kann.

Etwas spezieller formuliert es Prieto-Diaz: *„Unter Software Reuse versteht man die Verwendung existierender Software-Komponenten für die Erstellung neuer Software-Systeme.*[472]*“*

Griffel fordert Erweiterbarkeit bzw. Anpassung von bestehenden (auch geschlossenen) Systemen und begründet dies mit *„Wiederverwendung heißt aber auch Weiterverwendung*[473]*“*. Genau an dieser Stelle entstehen große und breite Diskussionen, um Sinn und Zweck verschiedener Arten der Wiederverwendung von Komponenten und dem eigentlichen Paradigma der komponentenorientierten Anwendungsentwicklung. In der objektorientierten Welt[474] werden White-Box- und Grey-Box-Ansätze bevorzugt. Dem stellt sich der ingenieurmäßige Ansatz entgegen, der eine strikte Black-Box-Lösung fordert. Eine solche reine Black-Box-Wiederverwendung ist besonders für dynamische Szenarien geeignet, da die lose Kopplung zu einer Reduktion der Abhängigkeiten zwischen den einzelnen Systemen (Komponenten) führt.

Beim ingenieurmäßigen Ansatz[475] wird auf Wiederverwendungsmethodiken der klassischen Ingenieursdisziplinen (Maschinenbau und Elektrotechnik) zurückgegriffen. In diesen Bereichen hat sich die Komponentenorientierung durch Standardisierung und Normung von (Bau-)Teilen und (Bau-)Gruppen durchgesetzt und bewährt. Ein weiteres Merkmal des ingenieurmäßigen Ansatzes ist die Industrialisierung (klassisch: Massenproduktion, Arbeitsteilung, Spezialisierung, Rationalisierung, Automatisierung, kontinuierliche Verbesserung, Standardisierung, Plattformstrategie, Modulkomponenten, Verringerung der Fertigungstiefe, Nutzung globaler Märkte, globale Nutzung von

---

[470] Biggerstaff; Perlis 1989.
[471] IEEE 1990b.
[472] Prieto-Diaz 1993, S. 61.
[473] Griffel 1998, S. 18.
[474] Vgl. Griffel 1998, S. 18.
[475] Vgl. Wedekind; Ortner 1980, S. 10.

Lohngefällen) des Entwicklungs- und Produktionsprozesses. In der Software-Entwicklung findet auch eine Industrialisierung[476] statt. Als Schlüssel gilt hierbei die Arbeitsteilung[477]. In der Literatur finden sich für einen solchen ingenieurmäßigen Ansatz auch häufig die Begriffe „Software Technik“[478] oder „Software Engineering“[479]. Balzert definiert „Software Technik“ als *„zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.*[480]*“*

Im ingenieurmäßigen Sinne beschreibt Ortner das Paradigma der komponentenorientierten Anwendungsentwicklung wie folgt: *„Die Komponenten werden aus einem Katalog entnommen und dann nach einem Plan (z. B. einer Syntax oder Grammatik, auf der Basis eines Frameworks oder mit Hilfe von Stücklisten und Konstruktionsplänen) zusammengesetzt.* [481]*“*

Dabei wird deutlich welcher Art ein „Software-Ingenieur“ sein muss: ein Sprachbau-Ingenieur[482]. Da alle Artefakte, die der „Software-Ingenieur“ verarbeitet und auch das Ergebnis selbst (das Softwaresystem) sprachliche Konstrukte sind, ist die Bezeichnung Sprachbauingenieur angemessen.

Mit dem Komponenten-Paradigma erfolgt ein Paradigmenwechsel: Weg von der Entwicklung monolithischer Standardsoftware-Pakete für Unternehmensfunktionen, die begrenzt an die Bedingungen ihres Einsatzbereiches angepasst werden können (Customizing) und manchmal auch organisationale Veränderungen im Unternehmen notwendig machen, hin zur Entwicklung individuell gestaltbarer, flexibler Anwendungen, die sich aus Komponenten zusammensetzen, die auf Märkten gehandelt werden.

Ein solches Paradigma zielt auf eine reine Black-Box-Wiederverwendung von Komponenten ab. Komponenten müssen also in verschiedenen Varianten vorliegen, die beim Bau eines Anwendungssystems ausgewählt werden können (siehe Abbildung 49). Damit werden unüberschaubare Versionen von Komponenten vermieden, die individuell angepasst wurden.

---

[476] Vgl. Taubner 2005.
[477] Vgl. Wedekind 2005.
[478] Vgl. z. B. Siedersleben 2003.
[479] Vgl. z. B. Sametinger 1997.
[480] Balzert 1989, S. 36.
[481] Ortner 2005a, S. 115.
[482] Vgl. Ortner 2002, S. 50.

	Anpassung	Auswahl
Systeme	Customizing	Standardsoftware verschiedener Hersteller
Komponenten	Versionen	Varianten

Abbildung 49: Anpassung im Vergleich zu Auswahl[483]

4.2.2 Komponentenmodelle

Auch bei dem Begriff „Komponentenmodell" liegt keine einheitliche Verwendung vor. Gruhn und Thiel[484] setzen den Begriff „Komponentenmodell" mit „Komponentensystem" gleich. Griffel[485] (und andere Autoren) verwenden die Begriffe „Komponentenmodell" und „Komponentenarchitektur" synonym. Hansen und Neumann[486] hingegen verwenden den Begriff „Komponententechniken".

Gruhn und Thiel definieren: *„Ein Komponentenmodell legt einen Rahmen für die Entwicklung und Ausführung von Komponenten fest, der strukturelle Anforderungen hinsichtlich Verknüpfungs- bzw. Kompositionsmöglichkeiten (Komposition) sowie verhaltensorientierte Anforderungen hinsichtlich Kollaborationsmöglichkeiten an die Komponenten stellt. Darüber hinaus wird durch ein Komponentenmodell eine Infrastruktur angeboten, die häufig benötigte Mechanismen wie Verteilung, Persistenz, Nachrichtenaustausch, Sicherheit und Versionierung implementieren kann.[487]"*.

Weinreich und Sametinger[488] definieren die Standardisierung von Komponentenimplementierung, Schnittstellen, Metadaten, Namensgebung, Interoperabilität, Anpassung, Zusammensetzung, Evolution und Installation als Elemente eines Komponentenmodells.

Nach Councill und Heinemann[489] ist ein Komponentenmodell ein Standard für Interaktion und Komposition. Eine Komponentenmodellimplementierung stellt eine Software

---

[483] In Anlehnung an Ortner 2005a, S. 116.
[484] Vgl. Gruhn; Thiel 2000.
[485] Vgl. Griffel 1998, S. 77ff.
[486] Vgl. Hansen; Neumann 2001, S. 996.
[487] Gruhn; Thiel 2000, S. 293.
[488] Vgl. Weinreich; Sametinger 2001, S. 37f.
[489] Vgl. Councill; Heinemann 2001, S. 7.

dar, die das Ausführen von (Software-)Komponenten unterstützt, die einem Komponentenmodell entsprechen. Die Standardisierung von Interaktion und Komposition bezieht sich dabei auf jeweils ein Komponentenmodell.

Zusammenfassend lässt sich ein Komponentenmodell als ein Modell zur Definition, Implementierung und Ausführung von Komponenten definieren.

In der Praxis haben sich vier verschiedene Komponentenmodelle etabliert: Das Component Object Model (COM) der Firma Microsoft[490], JavaBeans/Enterprise Java Beans (EJB) der Firma Sun[491], das CORBA Component Model (CCM) der Object Management Group[492] (OMG) sowie die Common Language Infrastructure (CLI), die sowohl von der European Computer Manufacturers Association[493] (ECMA) als auch von der International Standardization Organization[494] (ISO) standardisiert wurde. Diese vier bedeutenden Komponentenmodelle werden im Folgenden kurz vorgestellt. Umfangreiche Vergleiche finden sich bei Gruhn und Thiel[495], Longshaw[496] sowie Schwichtenberg[497]. Weitere Komponentenmodelle wie Voyager[498], das San Francisco-Framework[499] und Bonobo/GNOME Components[500] werden im Folgenden wegen der fehlenden Marktbedeutung (vgl. Schwichtenberg[501]) nicht weiter betrachtet.

4.2.2.1 Component Object Model (COM)

Das Component Object Model (COM) ist ein von der Firma Microsoft entwickeltes Komponentenmodell, das seine Ursprünge im Object Linking and Embedding (OLE) hat, einer Technik zur Komposition von unterschiedlichen Dokumentenbausteinen[502]. Der Begriff ActiveX wurde eine Zeit lang als Marketingbegriff von Microsoft mit COM gleichgesetzt. Durchgesetzt hat sich aber inzwischen, ActiveX als die GUI- und browserbasierten Teile von COM zu definieren[503].

---

[490] Vgl. Microsoft o. J.
[491] Vgl. Sun o. J.
[492] Vgl. OMG 2006.
[493] Vgl. ECMA 2005.
[494] Vgl. ISO 2005.
[495] Vgl. Gruhn; Thiel 2000.
[496] Vgl. Longshaw 2001.
[497] Vgl. Schwichtenberg 2003, S. 49-79.
[498] Vgl. Griffel 1998, S. 93ff.
[499] Vgl. Griffel 1998, S. 118ff.
[500] Vgl. Meeks 2001.
[501] Vgl. Schwichtenberg 2003, S. 80-82.
[502] Vgl. Gruhn; Thiel 2000, S. 48.
[503] Vgl. Griffel 1998, S. 80.

4.2.2.2 Java-Komponenten

Java ist eine Programmiersprache, die im Jahr 1995 von der Firma Sun erstmals angekündigt wurde[504]. Sie ist eine objektorientierte, plattformunabhängige Sprache, die für verschiedene Betriebssystem- und Prozessor-Plattformen verfügbar ist. In Zusammenhang mit Java wird in der Regel nur von zwei Komponentenarten gesprochen: Java Beans und Enterprise Java Beans[505]. Szyperski[506] und Bodoff et al.[507] identifizieren innerhalb des Java-Konzeptes jedoch fünf verschiedene Komponentenmodelle: Applets, JavaBeans, Enterprise Java Beans, Java Servlets und Application Client Components. Diese Aufzählung müsste noch um Midlets (Applets in J2ME, siehe Abschnitt 3.4.2) ergänzt werden. Engel, Koschel und Tritsch[508] dagegen unterscheiden Java-Komponenten nur in Java-Web-Komponenten (Java Server Pages, Servlets), Java Beans und Enterprise Java Beans. Enterprise Java Beans und Java Servlets sind die Kernbausteine der Java 2 Enterprise Edition (J2EE)[509].

4.2.2.3 CORBA Component Model (CCM)

Mit Version 3.0 wurde die zunächst nur auf verteilte Objektsysteme ausgerichtete Common Object Request Broker Architecture (CORBA) um ein Komponentenmodell erweitert, das CORBA Component Model (CCM) bzw. CORBA Components genannt wird[510]. Hierbei wurde das Ziel verfolgt, einige Beschränkungen von CORBA zu beseitigen[511]. Das CORBA Component Model basiert konzeptionell in vielen Bereichen auf dem Java Enterprise Beans Ansatz und erweitert diesen u. a. hinsichtlich der Unterstützung anderer Sprachen außer Java[512,513,514]. Szyperski stellt fest, dass das CORBA Component Model von Enterprise Java Beans inspiriert wurde und dieses selbst wieder von dem Component Object Model und den COM-Diensten (Microsoft Transaction Server, COM+)[515].

---

[504] Vgl. Deitel 1997.
[505] Vgl. Hansen; Neumann 2001, S. 970.
[506] Vgl. Szyperski; Gruntz; Murer 2002, S. 302.
[507] Vgl. Bodoff et al. 2002, S. 12.
[508] Vgl. Engel; Koschel; Tritsch 2002, S. 62.
[509] Vgl. Eicker; Schwichtenberg; Büning 2001, S. 639.
[510] Vgl. Szyperski; Gruntz; Murer 2002, S. 247.
[511] Vgl. Wang; Schmidt; O'Ryan 2001, S. 559-560.
[512] Vgl. Goedicke; Neumann; Zdun 2001, S. 126.
[513] Vgl. Szyperski; Gruntz; Murer 2002, S. 247.
[514] Vgl. Hansen; Neumann 2001, S. 973.
[515] Vgl. Szyperski; Gruntz; Murer 2002, S. 395.

4.2.2.4 Common Language Infrastructure (CLI)

Die Common Language Infrastructure (CLI) ist ein Komponentenmodell, das ursprünglich seit Ende der 1990er Jahre von der Firma Microsoft unter verschiedenen Namen (COM 3.0, Next Generation Windows Services, .NET Framework) als Nachfolger von COM entwickelt und von der ECMA (im Jahre 2001) und der ISO (im Jahre 2002) standardisiert wurde. Von der ECMA wurde die CLI im Dezember 2001 unter ECMA-335[516] standardisiert. Mit kleinen Änderungen wurde der Standard im Dezember 2002 von der ISO übernommen als ISO/IEC 23271[517]. Aktuell liegt der Standard bei der ECMA in der 4. Auflage seit Juni 2006 vor[518].

4.2.2.5 Zusammenfassende Übersicht

Im Vergleich verschiedener Komponentenmodelle zeigen sich sowohl erhebliche Unterschiede als auch Gemeinsamkeiten. Szyperski[519] sieht die Gemeinsamkeiten der Komponentenmodelle COM, CCM, JavaBeans/EJB und CLI/.NET insbesondere darin, dass sie alle folgende Mechanismen unterstützen: Kapselung, Transfer-Format für Komponenten, spätes Binden, dynamischer Polymorphismus, Ereignisse, Metadaten, Serialisierung und Persistenz.

Erhebliche Unterschiede gibt es bei den Komponentenmodellen hinsichtlich des Aufbaus der Komponenten (Komponentenbaupläne), der Namensgebung, der Beschreibung und des Aufrufs von Komponenten (Service Access-Protokolle), der Art der Installation von Komponenten sowie der Anzahl und Plattformbindung der verfügbaren Implementierung und den durch die Laufzeitumgebung bereitgestellten Diensten.

Als Service Access Protocol (SAP) bezeichnen Beitz et al.[520] ein Protokoll zur Nutzung eines Dienstes. In ihrer Arbeit unterscheiden sie zwischen Middleware Services (z. B. RPC, siehe Abschnitt 3.6.1) und Non-Middleware-Services (FTP, Telefon, Telefax). Die verschiedenen Komponentenmodelle nutzen unterschiedliche Service Access-Protokolle. Die Kopplung (d. h. die gegenseitige Nutzung) von homogenen (Software-)Komponenten erfolgt in der Regel auf Basis eines Binärstandards für lokale Aufrufe (Local Procedure Calls) oder eines Service Access-Protokolls für entfernte Aufrufe (Remote Procedure Call). Die Kopplung von (Software-)Komponenten auf

[516] ECMA 2005.
[517] Vgl. ISO 2005.
[518] ECMA 2006.
[519] Vgl. Szyperski; Gruntz; Murer 2002, S. 385.
[520] Vgl. Beitz; Bearman; Vogel 1995.

Basis verschiedener Komponentenmodelle ist möglich durch gemeinsame Service Access-Protokolle (z. B. Internet Inter ORB Protocol (IIOP) in CORBA und Java; SOAP in CORBA, COM, Java und .NET) oder sogenannter Component Bridges[521].

Tabelle 5 gibt einen Überblick über die wichtigsten Vergleichskriterien.

Merkmal	COM	Java	CCM	CLI/.NET
Standardisierungsgrad	Niedrig	Hoch	Hoch	Mittel
Betriebssystem-unabhängigkeit	Niedrig	Hoch	Hoch	Mittel
Sprachunabhängigkeit	Hoch	Keine	Mittel	Sehr hoch
Namensgebung	GUIDs	Hierarchische Textnamen	Hierarchische Textnamen	Hierarchische Textnamen
Explizite Schnittstellen	Ja	Ja	Ja	Ja
Metadaten	Möglich	Pflicht	Pflicht	Pflicht
Zentrale Registrierung der Komponenten	Ja	Ja (bei EJB und Servlets)	Ja	Nein
Container-Ausführung	Optional	Ja	Ja	Ja
Unterstützung für Application Server	Ja	Ja	Ja	Ja
Verwendung eines Application Servers	Optional	Optional, für EJB Pflicht	Pflicht	Optional
Asynchrone Komponentennutzung	Queued Components	Message-Driven Beans	Nein	Queued Components
Spezialisiertes Komponentenmodell für Webserver	Nein	Ja (Servlets)	Nein	Ja (ASP.NET)
Spezialisiertes Komponentenmodell für visuelle Komponenten	Ja (ActiveX Controls)	Ja (Applets, Java Beans)	Nein	Ja (Windows Forms Controls)
Dynamisches Binden	Ja (IDispatch)	Ja (Reflection)	Ja (Dynamic Invocation Interface)	Ja (Reflection)
Service Access-Protokolle für Out-Process-Components	DCOM, SOAP	RMI-JRMP, RMI-IIOP, SOAP	IIOP, SOAP	CLI-Remoting, SOAP
Interoperabilität	CLI/.NET	CCM	EJB	COM
Standardisierter Handel	Nein	Nein	Ja, COS-Trading	Nein

Tabelle 5: Zusammenfassender Vergleich der Komponentenmodelle[522]

Ein anderer, inzwischen aber veralteter, Vergleich findet sich in Griffel[523].

[521] Vgl. Geraghty et al. 1999, S. 29ff.
[522] In Anlehnung an Schwichtenberg 2003, S. 82-83.
[523] Vgl. Griffel 1998, S. 97.

4.2.3 (Software-)Komponenten-Handel

Handel ist ein Überbegriff für den wirtschaftlichen Austausch von Bedürfnis und Bedürfnisdeckung. Hinsichtlich der verschiedenen Arten reicht der Handel von einem privaten Tausch bis hin zur marktlichen Koordination. Die marktliche Koordination beschreibt das kumulierte Aufeinandertreffen von Nachfrage (Bedürfnis) und Angebot (Problemlösung), den Tausch von Gütern bzw. die Abstimmung einer ganzen Reihe wirtschaftlicher Aktivitäten, wobei hierfür der Begriff „Markt“ im Allgemeinen verwendet wird[524].

Mit der Verfügbarkeit weltweiter Daten- und Informationsnetze wird eine neue Marktform, der virtuelle Handel, ermöglicht[525]. Beim virtuellen Handel existiert kein Marktplatz, als realer Ort der physischen Begegnung von Anbietern und Nachfragern. Im Sinne einer semiotischen Virtualität (siehe Abschnitt 3.1.3.1) ist der Marktplatz nur virtuell präsent. Es handelt sich hierbei um einen „virtuellen Raum“, der nur mit Hilfe elektronischer Geräte bzw. dem Einsatz von Informationstechnologie von allen Teilnehmern „betreten“ werden kann[526]. In diesem Sinne müssen die Marktteilnehmer zum Marktplatz-Zugangsgerät (z. B. PC, der an das Internet angeschlossen ist) hingehen. Anbieter und Nachfrager „treffen“ sich unabhängig von Ort und Zeit auf einem virtuellen Marktplatz, der alle notwendigen Informationen für sie bereitstellt.

Ein mobiler Marktplatz ist ein elektronischer Marktplatz, der Dienstmobilität und Benutzermobilität unterstützt[527]. Im Sinne der Dienstmobilität muss der Marktplatz eindeutig adressierbar sein, damit er unabhängig vom Ort des Anwenders erreichbar ist. Zur Unterstützung der Benutzermobilität sollte ein mobiler Marktplatz den Zugang durch mobile Endgeräte unterstützen. Dafür müssen drahtlose Kommunikationsnetze unterstützt und z. B. in Form einer WAP-Schnittstelle bereitgestellt werden. Das mobile Marktplatzsystem selbst ist nicht notwendigerweise physisch mobil im Sinne der Endgerätemobilität, sondern wird üblicherweise auf einem oder mehreren verteilten Servern (nomadisch verteiltes System) betrieben. Der Vorteil des mobilen Marktplatzes ist, dass ein Marktteilnehmer nicht mehr zum Markt hingehen muss, sondern der Markt stattdessen zum Nachfrager bzw. auf dessen (mobiles) Zugangsgerät kommt, unabhängig vom Aufenthaltsort des Nachfragers.

---

[524] Vgl. Kollmann 2001, S. 1.
[525] Vgl. Kollmann 2001, S. 6.
[526] Vgl. Kollmann 2001, S. 39.
[527] Vgl. Lonthoff 2004, S. 454.

Der Handel auf elektronischen Marktplätzen basiert im Wesentlichen auf Informationsaustausch. Information ist ein zentraler Wettbewerbsfaktor[528], insbesondere auch, um die notwendige Aufmerksamkeit bei potenziellen Kunden zu erzeugen[529]. Informationen beeinflussen die grundlegenden Dimensionen des Wirtschaftens auf drei Ebenen[530]: Informationsgewinnung, -verarbeitung und -übertragung. Da Menschen beschränkte Fähigkeiten bezüglich der Speicherung, Nutzung und Verarbeitung von Informationen haben[531], liegt die Aufgabe des virtuellen Handels besonders in der Informationsverarbeitung. Information ist hierbei gerichtet[532]. „Pull" bezeichnet dabei das aktive Beschaffen von Informationen. „Push" hingegen bezeichnet den passiven Erhalt von Informationen. Hanser[533] beschreibt eine Entwicklung weg von der reinen Push-Information, hin zu einer Push-Pull-Information im Sinne einer Interaktion. Für derartige Interaktion gibt es bspw. Publish/Subscribe-Mechanismen[534].

Im Umfeld elektronischer Märkte wird häufig vom Markt mit vollständiger Konkurrenz gesprochen[535]. Damit sind folgende Eigenschaften verbunden: atomistische Angebots- und Nachfragestruktur, Fehlen von Präferenzen bzw. Diskriminierungen von Anbietern und Nachfragern, vollständige Markt- bzw. Preistransparenz, unbegrenzte Mobilität sämtlicher Produktionsfaktoren und Güter[536]. Elektronische Marktplätze erfüllen sicherlich nicht das Ideal der vollständigen Konkurrenz, können sich diesem im Vergleich zu herkömmlichen Märkten jedoch weitaus besser nähern: die Markt- und Preistransparenz ist größer, jeder kann sich schneller informieren, durch die Möglichkeit des anonymen oder pseudonymen Auftretens im Netz können Diskriminierungen vermieden werden. Gerade mit fortschreitender Durchdringung des Alltags mit Technik wird auch der Zugang zu elektronischen Marktplätzen immer leichter. Die vollständige Markt- und Preistransparenz kann durch elektronische Marktplätze aber (noch) nicht realisiert werden, da durch die unüberschaubare Flut gehandelter Gegenstände häufig die Vergleichbarkeit der Handelsgüter nur schwer gelingt. Außerdem werden Marktformen unterstützt, bei denen die Preisermittlung unklar bzw. verdeckt erfolgt.

---

[528] Vgl. Kollmann 2001, S. 10.
[529] Vgl. Davenport; Beck 2001, S. 3.
[530] Vgl. Kollmann 1998, S. 44f.
[531] Vgl. Noam 1997, S. 36.
[532] Vgl. Kollmann 2001, S. 27.
[533] Vgl. Hanser 1995, S. 36f.
[534] Vgl. Lehner 2002, S. 27f.
[535] Vgl. Merz 2002, S. 98.
[536] Vgl. Kollmann 2001, S. 41.

Ein elektronisches Marktplatzsystem ist ein vollständiges System (siehe Abbildung 50), das Nachfragern und Anbietern ermöglicht, alle notwendigen Handelstransaktionen abzuschließen[537]. Dabei lassen sich grob drei Phasen einer Handelstransaktion identifizieren[538]: Informations-, Verhandlungs- und Abwicklungsphase.

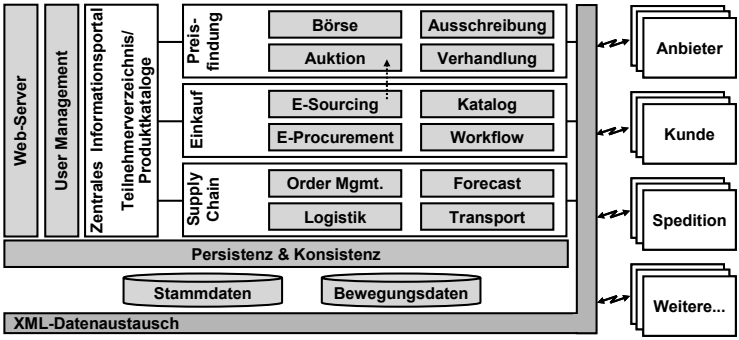

Abbildung 50: Komponenten eines elektronischen Marktplatzsystems[539]

In der Informationsphase möchten Nachfrager bzw. Anbieter herausfinden, welche Produkte bzw. Leistungen existieren. Hierzu werden die Spezifikation der Produkte, die Bezugsquellen und die Konditionen benötigt. Auch gesamtwirtschaftliche Rahmeninformationen, Brancheninformationen oder Trends können bei der Suche relevant sein. Resultate der Informationsphase sind bspw. Listen interessanter Marktpartner und Angebote bzw. Nachfragen.

Während der Vereinbarungsphase wird mit den Transaktionspartnern Kontakt aufgenommen. Die Transaktionskonditionen werden mit den gewählten Marktpartnern vereinbart, wie bspw. Zahlungsbedingungen, Garantieleistungen oder Serviceleistungen. Resultate dieser Phase sind Vorraussetzung und rechtliche Grundlagen der Transaktion.

Die Abwicklungsphase bezeichnet die Phase, in der die eigentliche Transaktion durchgeführt wird. Bei physischen Gütern erfolgen während dieser Phase die Verpackung und Lieferung der bestellten Produkte. Beim Handel von (Software-)Komponenten

[537] Vgl. Merz 2002, S. 20f.
[538] Vgl. Schmid 1993, S. 467f.
[539] In Anlehnung an Merz 2002, S. 813.

wird in dieser Phase dem Nachfrager die bestellte (Software-)Komponente bereitgestellt, z. B. mittels Dateitransfer oder Nutzung der (Software-)Komponente als Dienst.

„Virtuelle Marktplätze setzen ein Umfeld mit elektronisch handelbaren bzw. beschreibbaren Gütern voraus, bei denen sinnliche Eindrücke (schmecken/riechen/fühlen) eher sekundär sind, so dass die Marktprozesse […] gut multimedial adaptiert werden können. [540]*“* Eine geforderte Eigenschaft von (Software-)Komponenten ist ihre Vermarktbarkeit (siehe Abschnitt 4.2.1.1). Komponenten sind abgeschlossene wirtschaftliche Güter, die auf Märkten gehandelt werden können.

Um vollständig (alle Marktphasen betreffend) virtuell handelbar zu sein, muss die Eigenschaft eines digitalisierbaren Guts vorliegen. (Software-)Komponenten sind vollständig digitalisierbar. Sie sind immateriell und bedürfen keiner physischen Gestalt („Form“). Dies ist wiederum auch ein Grundproblem von (Software-)Komponenten. Sie sind immateriell (intangible), also „unsichtbar“, jedoch durch Code (physisch) repräsentiert aber manchmal nur binär als Black-Box. Zudem sind (Software-)Komponenten hoch komplex und erklärungsbedürftig.

Daher ist eine Grundvoraussetzung für die Entstehung eines Komponenten-Handels, dass Komponenten vollständig beschrieben werden können, z. B. mit Hilfe einer (einheitlichen) Spezifikation bzw. eines Beschreibungsrahmens, wie er z. B. von der GI vorgeschlagen wird[541]. Eine solche Beschreibung (Spezifikation) erfolgt sprachlich. Hierbei ist die Sprache das Bindeglied zwischen Mensch und Technik (Hardware und Software). Da es um Sprache geht, entspricht der Komponenten-Handel einem „Schema-Handel“. Denn eine Komponente ist ein Schema einer konkreten Instanz und mittels (Meta-)Schemata (Beschreibung) muss eine Komponente auf der Metaebene vollständig spezifizierbar sein.

Szyperski stellt fest, dass es in Unternehmen weit mehr selbst entwickelte Java-Komponenten als käufliche Standard-Komponenten gibt[542]. Untersuchungen zur Verbreitung von (Software-)Komponenten (Cutter Consortium[543], ComponentSource[544], Schwichtenberg[545]) zeigen, dass es einen heterogenen Markt für (Soft-

[540] Kollmann 2001, S. 66.
[541] Vgl. Ackermann et al. 2002.
[542] Vgl. Szyperski; Gruntz; Murer 2002, S. 387.
[543] Vgl. Harmon 2000.
[544] Vgl. ComponentSource o. J.

ware-)Komponenten gibt und ein Bestehen auch in Zukunft zu erwarten ist. Für den Komponenten-Handel ergibt sich daraus die Konsequenz, dass heterogene (Software-)Komponenten unterstützt werden müssen.

Beispiele für existierende (Software-)Komponenten-Marktplätze sind SourceForge, Software-Marktplatz und Component-Source[546,547]. Im Rahmen einer Anforderungsanalyse[548] hat Schwichtenberg[549] eine Anforderungsspezifikation an den Handel heterogener (Software-)Komponenten (Schwichtenberg nennt dies Component Trading) formuliert.

Das zentrale Problem des Komponenten-Handels liegt im Finden geeigneter, möglichst optimaler Komponenten. Bei der Lösung dieses Problems muss das rechnerunterstützte virtuelle Handelssystem eine geeignete Unterstützung bereitstellen. Am bequemsten wäre ein vollautomatischer Ablauf der Komponentenbeschaffung (automatisiertes externes Anwendungsmanagement). Entsprechende Mechanismen dieser Übereinstimmungssuche werden in der Arbeit von Hümmer[550] mit Hilfe von „Match Making-Algorithmen" vorgestellt. Zur elektronischen Unterstützung der Vertragsverhandlung im Allgemeinen gibt das Sonderheft des Electronic Markets Journal[551] einen guten Überblick möglicher Lösungsansätze. Genau aus dieser Unterstützungsfunktion ergibt sich ein Mehrwert beim Zwischenschalten eines Intermediärs zwischen Anbieter und Nachfrager. Mögliche Geschäftsmodelle werden im Abschnitt 4.3 vorgestellt.

Eine weitere Motivation für den Komponenten-Handel ist die explizite Wiederverwendung i. S. einer Industrialisierung der Software-Entwicklung (Arbeitsteilung verstanden als Problemaufteilung[552]). In dem Beitrag von Ravichandran und Rothenberger[553] werden Komponenten-Marktplätze, die (Software-)Komponenten nach dem Black-Box-Wiederverwendungsprinzip handeln, als die beste und auch wahrscheinlichste Lösung der Software-Wiederverwendung herausgestellt. Solche Komponenten-Marktplätze können die komponentenorientierte Software-Entwicklung und generell

[545] Vgl. Schwichtenberg 2003, S. 80-82.
[546] Vgl. Hau; Mertens 2002, S. 337.
[547] Vgl. Fettke; Loos; Viehweger 2003, S. 5.
[548] Vgl. Hansen; Neumann 2005, S. 253.
[549] Schwichtenberg 2003.
[550] Hümmer 2004.
[551] Zimmermann et al. 2006.
[552] Vgl. Parnas 1972.
[553] Vgl. Ravichandran; Rothenberger 2003.

komponentenorientierte Anwendungssysteme vorantreiben. Sollen handelbare (Software-)Komponenten entwickelt werden, muss ihre externe Wiederverwendung das (Entwicklungs-)Ziel sein. Somit mündet das Prinzip der Wiederverwendung in den Komponenten-Handel auf entsprechenden Marktplätzen.

Im Kapitel 5 wird ein implementiertes Marktplatzsystem für den Handel mit (Software-)Komponenten für das mobile Umfeld auf Basis des Konzepts von Abschnitt 4.5 vorgestellt.

4.3 Geschäftsmodelle für das externe Anwendungsmanagement

Eine ökonomische Voraussetzung für die Existenz eines Marktes ist ein Handel im Sinne eines Tausches zwischen Anbieter und Nachfrager zum Ausgleich ungleich verteilter Ressourcen, Produkte und Dienstleistungen[554]. Für die Entwicklung möglicher Geschäftsmodelle muss die Frage beantwortet werden, wer Nachfrager und wer Anbieter ist und welche Rolle der Intermediär spielt.

Mögliche Anbieter hierbei sind Entwickler, Software-Hersteller, Komponenten-Hersteller, Open-Source-Projekte, Software-Dienstleister und Komponenten-Händler (Wiederverkäufer). Mögliche Nachfrager können Entwickler, Open-Source-Projekte, Anwender, Konsumenten und Komponentenhändler sein.

Sarkar, Butler und Steinfield[555] stellen in ihrer grundlegenden Arbeit über *„Intermediaries and Cybermediaries“* mögliche Rollen von Intermediären auf elektronischen Marktplätzen vor: Vermittlung zwischen Anbieter und Nachfrager (Broker), klassischer Handel von einzelnen Komponenten, Handel und Komposition geeigneter Komponenten, Nutzung von Komponenten als Dienst des Marktplatzes. Innerhalb der verschiedenen Rollen des Marktplatzes stehen noch unterschiedliche Marktplatzformen zur Auswahl, wie bspw. Auktionen, Kataloge, Börsen und schwarze Bretter[556]. Zusätzlich kann unabhängig von der Marktplatzform noch ein Dienst zum (geeigneten) Zusammenstellen von Komponenten angeboten werden[557]. Dieser Dienst würde Teilaufgaben des internen Anwendungsmanagements erfüllen und dem Kunden eine zusammengesetzte Komponente gröberer Granularität bereitstellen. Weiterhin ist denkbar,

[554] Vgl. Wilmsen1972 zitiert in Schmid 1993, S. 1.
[555] Vgl. Sarkar; Butler; Steinfield 1995.
[556] Vgl. Kollmann 2001, S. 85-89.
[557] Vgl. Szyperski; Gruntz; Murer 2002, S. 17.

dass ein Marktplatz vollständig die Aufgabe des Zusammenstellens eines Anwendungssystems aus Komponenten übernimmt und der Nachfrager dann dieses individuell zusammengestellte Anwendungssystem vollständig bezieht oder als Dienst nutzt, wie z. B. beim Aplication Service Providing (ASP).

Die Wettbewerber eines Komponenten-Marktplatzes sind die etablierten Anbieter aus der Anwendungsentwicklungsbranche, wie z. B. Systemhäuser (zur Entwicklung von Individuallösungen), Standardsoftwareanbieter, Verzeichnisanbieter[558], Berater oder selbstständige Entwickler.

4.3.1 Nutzungsarten von (Software-)Komponenten-Marktplätzen

Je nach Rolle des Marktplatzes können verschiedene Nutzungsarten bzw. Betriebszwecke eines Komponenten-Marktplatzes unterschieden werden. Sarkar et al.[559] konnten zehn grundlegende Funktionen von Marktplatzbetreibern identifizieren: Search and Evaluation, Needs Assessment and Product Matching, Customer Risk Management, Product Distribution, Product Information Dissemination, Purchase Influence, Provision of Customer Information, Producer Risk Management, Transaction Economies of Scale und Integration of Consumer and Producer Needs.

Eine weitere Unterscheidung kann im Hinblick auf den Betriebszweck von Komponenten-Marktplätzen getroffen werden. Hierbei kann zwischen Outsourcing, Gebrauch, Handel, Entwicklung und Offshoring unterschieden werden (siehe Abbildung 51).

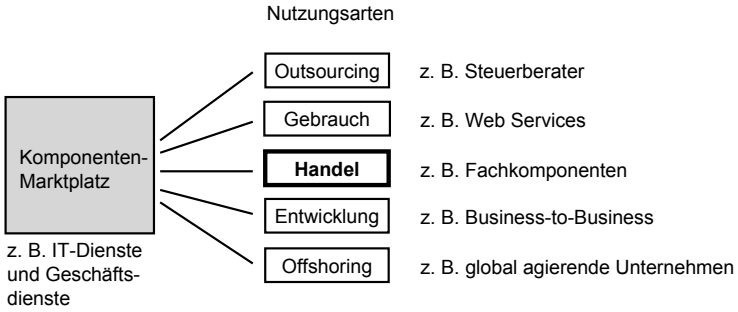

Abbildung 51: Nutzungsarten von (Software-)Komponenten-Marktplätzen

[558] Vgl. Traas; Hillegersberg 2000, S. 114.
[559] Vgl. Sarkar; Butler; Steinfield 1995.

Im Folgenden werden Betreibermodelle nach Nutzungsarten des Komponenten-Marktplatzes vorgestellt. Im Sinne einer Komponenten-Beschaffung bzw. -Bereitstellung umfassen alle hier vorgestellten Nutzungsarten Funktionen des externen Anwendungsmanagements (siehe Abschnitt 2.5.1).

Outsourcing

Es wird eine komplette betriebsnotwendige Funktion als Anwendungssystem bei einem externen Unternehmen ausgelagert. Über Service Level Agreements (SLAs) wird der Leistungsumfang vertraglich festgelegt. Das Management der durch Outsourcing ausgelagerten Funktion geht auf den Outsourcer über. Als Beispiel sei der Steuerberater als Anbieter von Buchhaltungs- und Rechnungswesendienste für Mandanten genannt.

Gebrauch

In diesem Szenario wird keine Kopie der Komponente übertragen (i. S. einer logischen Wiederverwendung von Komponenten), sondern Nachfrager erhalten die Möglichkeit eine Komponente als Dienst des Marktplatzes (i. S. einer physischen Wiederverwendung von Komponenten) zu nutzen. Über die klar definierten Schnittstellen werden entsprechende Eingaben zum Marktplatz gesendet und evtl. anfallende Ausgaben werden vom Marktplatz zum Nachfrager gesendet. Hierbei wird nur die Funktion einer Komponente genutzt. Der Marktplatzbetreiber sorgt für eine entsprechende Laufzeitumgebung für die Komponenten. Ein Beispiel für „Gebrauch“ sind Web Services, die über einen elektronischen Marktplatz mehrfach genutzt werden können.

Handel

Dies ist der klassische Ansatz des Handels mit Komponenten. Hier können direkt einzelne oder auch schon zusammengesetzte bzw. vorkonfigurierte Komponenten entweder direkt vom Marktplatz erworben werden oder es erfolgt eine Vermittlung zu Angeboten entsprechender Hersteller. Eine Möglichkeit ist der Handel von Fachkomponenten für die Entwicklung betrieblicher Anwendungssysteme. Aber auch eine Ausdehnung des Handels in den Bereich „Konsumentenkomponenten“ ist vorstellbar. Durch eine weite Verbreitung von Breitbandzugängen (DSL, UMTS etc.) steigt die Akzeptanz virtuellen Handels mit (Software-)Komponenten auch bei privaten Anwendern (Konsumenten). Nur ist dabei insbesondere auf die Vorkenntnisse des Konsumenten zu achten. Konfiguration und Installation müssen möglichst ohne Mithilfe des Konsumenten im Hintergrund (automatisiertes Anwendungsmanagement) ablaufen.

Entwicklung

Ein Komponenten-Marktplatz kann für Entwickler auch als eine (kommerzielle) Version eines Entwicklungs-Repositoriums verwendet werden. Dies ist vergleichbar mit der Nutzung von Software-Bibliotheken. Ein beispielhaftes Business-to-Business Szenario besteht im Aufbau und Betrieb bzw. Management zwischenbetrieblicher Anwendungen.

Offshoring

Ein Zusatzdienst für einen Marktplatz mit der Nutzungsart Entwicklung könnte z. B. nach erfolglosem Suchen einer entsprechenden Komponente darin bestehen, über Mechanismen, wie z. B. Publish/Subscribe[560] Anfragen bzw. Ausschreibungen für Komponenten-Hersteller zu hinterlegen. Mit Offshoring ist hierbei Auftragsentwicklung gemeint im Sinne von Mertens[561]: Wenn die Spezifikation (der Aufgabe) klar ist, dann verlagert sich die Entwicklung ins Ausland (z. B. Indien). Als Beispiel gelten global agierende Unternehmen, die zum einen Spezifikationen im Sinne der Arbeitsteilung nach außen vergeben und zum anderen Implementierungen weltweit vertreiben.

4.3.2 Kalkulationsmodelle

Auf Seiten der Komponenten-Anbieter können zur Preisberechnung ressourcenabhängige Größen, wie bspw. Anzahl der Komponenten oder Anzahl der erfüllten Aufträge angewendet werden. Auch volumenabhängige Größen, wie bspw. Lines of Code (LOC) oder Anzahl der Methoden können zur Preisbestimmung beitragen. Eine nutzerabhängige Bepreisung könnte in Abhängigkeit der Anzahl beteiligter Entwickler erfolgen.

Die anfallenden Kosten für Nutzer des Komponenten-Marktplatzes setzen sich im Wesentlichen aus Hardwarekosten, Personalkosten und Lizenzkosten für die bezogenen Marktplatzleistungen (Komponenten bzw. deren Gebrauch) zusammen.

Bei den Lizenzkosten für Software gibt es grundsätzlich drei Abrechnungsmethoden[562]. Die sogenannten „Named User“, bei denen nach bestimmten, durch Namen gekennzeichnete, Benutzern abgerechnet wird. Daneben werden bei den „Concurrent User“ die Lizenzgebühren für die Anwender bezahlt, die eine Software-Anwendung

[560] Vgl. Lehner 2002.
[561] Vgl. Mertens 2005b.
[562] Vgl. Grohmann 2002, S. 73.

gleichzeitig nutzen. Eine ressourcenabhängige Abrechnung wird bei der Erhebung der Lizenzgebühr z. B. nach der Anzahl der Prozessoren oder der Anzahl der genutzten Komponenten erreicht. Dieses aus dem Application Service Providing (ASP)-Umfeld stammende Abrechnungsmodell ist auf das Angebot von Komponenten-Marktplätzen übertragbar.

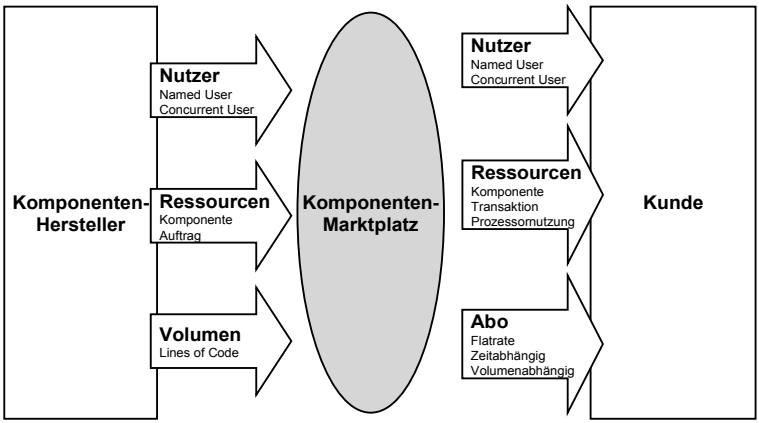

Abbildung 52: Preismodell für Komponenten-Marktplätze[563]

Je nach Anwendungsfall könnten Preismodelle für den Komponenten-Marktplatz nutzerorientiert als Named User (bspw. pro Account) bzw. Concurrent User, ressourcenorientiert, nach Anzahl der Prozessornutzung, Speichernutzung, Komponentenbezug oder Transaktionsanzahl, zeitorientiert, volumenorientiert oder pro System (bspw. Abonement oder Flatrate) konstruiert werden. Eine Mischform aus Anzahl genutzter Komponenten und Nutzungsdauer ist ebenfalls denkbar. Abbildung 52 fasst ein solches aus dem ASP-Umfeld angelehntes Preismodell zusammen.

Die Nutzung eines Komponenten-Marktplatzes in der Form des „Gebrauchs" hat dabei positive bilanzielle Auswirkungen für den Nachfrager. Ähnlich, wie die Nutzung von Mietsoftware, wird mit ASP-Abrechnungsmodellen Kostenreduktion durch sinkende Total Cost of Ownership (TCO), sinkender Hardwarebedarf und sinkender Bedarf an IT-Fachpersonal erreicht[564].

[563] In Anlehnung an Grohmann 2002, S. 74.
[564] Vgl. Grohmann 2002, S. 192-196.

TCO-Kosten sinken

Die Kostenkontrolle durch eine monatliche Abrechnung folgt dem „Total Cost of Ownership“ (TCO)-Prinzip[565]. Das TCO-Modell hilft IT-Verantwortlichen, die direkten und indirekten Kosten eines EDV-Systems oder einer Software-Anwendung zu kalkulieren. TCO umfasst unter anderem Kaufpreis, Upgrades, Schulungen und Administration. Bei dem Betriebszweck „Gebrauch“ des Marktplatzes wird das Anwendungssystem ähnlich einer ASP-Lösung fremdbezogen genutzt. Dabei werden die TCO transparenter und erheblich geringer[566]. Da die Infrastruktur in einem ASP-Modell mehrfach genutzt wird (engl. shared services), stehen auch kleineren Unternehmen leistungsfähige Systeme zur Verfügung, deren Erwerb zu kostenintensiv wäre. Gleiches gilt für Anwendungen, die für abgestimmte Kommunikation und Nutzung zwischen Unternehmen, Kunden und Lieferanten genutzt werden.

Hardware-Anforderungen sinken

Die Anforderungen an die Hardware bei der Nutzung von Komponenten-Diensten sinken. Die Computer der Kunden müssen über einen Netzwerkzugang (als Marktplatzzugang) verfügen. Für die Laufzeitumgebung der genutzten Komponente wird keine eigene Hardware benötigt. Dadurch können Server eingespart werden, die ansonsten die Ausführung der Komponenten sicherstellen müssten. Bei einer Komplettlösung, bei der das ganze Anwendungssystem als Dienst über den Marktplatz bereitgestellt wird, entfallen darüber hinaus auch die Kosten für besondere Schutzmaßnahmen zum Betrieb bspw. eines eigenen Rechenzentrums.

Bedarf an IT-Fachpersonal sinkt

Innerhalb des Unternehmens sinkt der Bedarf an IT-Fachpersonal, da Servicefunktionen vom Marktplatzbetreiber erbracht werden. Neue Versionen von Komponenten können entweder per Update aktualisiert oder zusätzlich zum Erwerb angeboten werden. Ein kosten- und zeitintensives Upgrade auf vielen einzelnen Rechnern kann durch Automatisierung eines Komponentenupdates vermieden werden. Durch geeignete Lizenzmodelle kann eine hohe Flexibilität der Abrechnung lizenzpflichtiger Anwender erreicht werden. Desweiteren können auch flexible Mitarbeitermodelle durch solche Lizenzmodelle unterstützt werden. Da die genutzten Komponenten vom Marktplatz aufgerufen werden, haben auch Telearbeiter und Außendienstmitarbeiter jederzeit und überall Zugriff auf die Komponenten.

---

[565] Vgl. Wild; Herges 2000.
[566] Vgl. Grohmann 2002, S. 194.

Zusammengefasst beschleunigen die Einsparungen den Return on Investment (ROI). Bei dem Erwerb von Softwarelizenzen sind nicht nur die TCO höher, sondern die tatsächlichen Kosten sind zudem auch schwerer zu kalkulieren. Neben Lizenzgebühren, Aufwendungen für Hardware und den laufenden Betrieb sind zwangsläufig unabsehbare Kosten für die Anpassungen, Schulungen und Produktivitätslücken in Zusammenhang mit IT-Problemen zu beachten. So wird durch den Einsatz von Komponenten-Marktplätzen eine Verkürzung der Bilanz auf der Aktiva Seite erreicht. Das Anlagevermögen wird im Bereich der IT-Infrastruktur geringer und die Kosten für internes und externes Personal werden sinken.

4.4 Anforderungen an ein mobiles Marktplatzsystem für den Komponenten-Handel

Herkömmliche elektronische Marktplatzsysteme[567,568], wie bspw. Component-Source oder Software-Marktplatz, sind nicht für den mobilen Einsatz konzipiert. Erst durch die Berücksichtigung der Anforderungen des mobilen Umfeldes an das Marktplatzsystem kann das System so gestaltet werden, dass ein daraus entstehendes elektronisches Marktplatzsystem im mobilen Einsatz gebrauchstauglich ist. Somit ergeben sich die Anforderungen eines mobilen Marktplatzsystems aus den Anforderungen eines elektronischen Marktplatzsystems unter besonderer Berücksichtigung des Aspekts Mobilität.

Durch Mobilität (siehe auch Roth[569], Hansmann et al.[570] und Lehner[571] und Abschnitt 3.1.3) entstehen auf zwei Seiten Anforderungen an das Marktplatzsystem: zum einen die Anforderungen, die sich aufgrund der Verwendung mobiler Endgeräte als Zugang zum Marktplatzsystem ergeben, und zum anderen die Anforderungen, die sich aufgrund mobiler Kommunikation ergeben.

Damit das zu konzipierende Marktplatzsystem als Ausprägung eines externen Anwendungsmanagements genutzt werden kann, kommt noch eine technische Anforderung hinzu. Das Marktplatzsystem muss zum Anbinden an andere Informations- und Kommunikationskomponenten klar definierte und offene Schnittstellen bereitstellen sowie

---

[567] Vgl. Traas; Hillegersberg 2000, S. 116.
[568] Vgl. Hau; Mertens 2002, S. 337.
[569] Vgl. Roth 2002.
[570] Vgl. Hansmann et al. 2001.
[571] Vgl. Lehner 2003.

seine Funktionalität in einem Verzeichnis registrieren. Da ein elektronisches Marktplatzsystem alle Phasen einer Handelstransaktion unterstützen soll, muss dieses nach Usability-Aspekten gebrauchstauglich gestaltet sein. Für die grundlegenden Funktionen des Marktplatzsystems werden Kataloge für die eingestellten Komponenten und Profile für Kunden bzw. Unternehmen benötigt.

Aus diesen im Folgenden angeführten Anforderungen wird in Abschnitt 4.5 ein Referenzmodell für ein mobiles Komponenten-Marktplatzsystem hergeleitet. Auf dieser Grundlage können Marktplatzssysteme schon beim Entwurf „mobil gemacht" werden, und es kann auf eine zentrale Middleware verzichtet werden. In Kapitel 5 wird mobiCOMP, eine prototypische auf Teilbereiche beschränkte Realisierung der Referenz-Marktplatzarchitektur, vorgestellt.

4.4.1 Usability

In diesem Abschnitt werden Anforderungen an die Gebrauchstauglichkeit (Usability) von Anwendungssystemen im Allgemeinen aufgestellt. Diese Anforderungen gelten für alle Menschen, also auch für den Anwender der auf den Marktplatz zugreifen soll. Die Anforderungen sind der europäischen Norm für die Arbeit mit Bildschirmgeräten[572] (ISO 9241 bzw. DIN 29241, Teil 10) entnommen. Teilweise können die Aussagen nur bedingt auf mobile Endgeräte übertragen werden, da einige mobile Endgeräte keinen Bildschirm haben. Die sieben in der Norm verzeichneten Anforderungen lauten im Einzelnen:

- *Aufgabenangemessenheit*
  Es muss die Frage geklärt werden, ob eine Benutzungsoberfläche mit Dialog- bzw. Formularfeldern für eine bestimmte Aufgabe angemessen ist. Evtl. kann die Aufgabe auch im Hintergrund ohne Benutzerinteraktion erfolgen.

- *Selbstbeschreibungsfähigkeit*
  Jede Interaktion muss unmittelbar verständlich oder zumindest auf „Knopfdruck" erklärbar sein. Die Selbstbeschreibungsfähigkeit impliziert ein kontextabhängiges, weitgehend in Umgangssprache (Gebrauchssprache) verfasstes Hilfesystem.

[572] ISO 1998.

- *Steuerbarkeit*
  Der Anwender muss in der Lage sein, jeden Schritt der Anwendung in Geschwindigkeit und Reihenfolge – soweit es hier Alternativen gibt – selbst zu beeinflussen.

- *Erwartungskonformität*
  Die Anwendung muss allgemeinen Konventionen der Kommunikation, Erfahrungen aus den bisherigen Arbeitsabläufen des Systems und der Ausbildung des Benutzers Rechnung tragen.

- *Fehlertoleranz*
  Das System muss Fehler bei der Eingabe möglichst vermeiden oder zumindest auffangen. Das Arbeitsergebnis sollte trotz fehlerhafter Eingabe noch ohne großen Korrekturaufwand erreichbar sein.

- *Individualisierbarkeit*
  Durch individuelles Gestalten der Benutzungsoberfläche soll der Anwender das System seinen eigenen Wünschen, Fähigkeiten und Arbeitserfordernissen anpassen können.

- *Lernförderlichkeit*
  Das System soll dem Anwender in einer Lernphase, soweit diese für die Nutzung der Anwendung nötig ist, Unterstützung und Anleitung zukommen lassen.

Durch die Beachtung oben genannter Kriterien wird eine Oberfläche bzw. ein Dialogsystem nicht unbedingt „schön“ – es lässt aber genügend Raum für das kreative Gestalten. Zur Individualisierbarkeit kann das Angebot auf Anwendergruppen bedarfsgerecht zugeschnitten werden. Ein weiterer Anpassungsschritt gelingt durch Personalisierung, also der Anpassung des Angebots auf einzelne Personen[573]. Dabei erfordern Individualisierung und Personalisierung das Sammeln und Speichern von Profildaten. Der Anspruch auf Selbstbeschreibungsfähigkeit und Erwartungskonformität stellt auch Bedingungen an die Suche.

Bei der Entwicklung eines elektronischen Marktplatzsystems für den Gebrauch mit mobilen Endgeräten darf nicht der Fehler gemacht werden, anzunehmen, dass der An-

[573] Vgl. Brandt; Lonthoff 2004, S. 39.

wender des Endgerätes automatisch ein Experte für mobile Anwendungssysteme sei. Der Hauptanteil der Anwender, die mobile rechnerunterstütze Arbeit leisten, sind keine Fachanwender (i. S. von IT-Experten), sondern bspw. Außendienstmitarbeiter, mobiler Pflegedienst oder Konsumenten. Aus diesem Grund ist es wichtig, die Gebrauchstauglichkeit der Software und somit das Benutzungskonzept mit in die Softwareentwicklung einzubinden.

4.4.2 Komponenten-Repositorium

Ein Komponenten-Repositorium ist eine Datenbank für die Ablage und das schnelle Auffinden wieder verwendbarer Komponenten[574]. Ein solches Komponenten-Repositorium beinhaltet die Komponenten mit allen relevanten Informationen über die Komponente, wie Entwurf, Historie, Verbindungen zu anderen Komponenten, Klassifikationen, Beschreibungen und Dokumentation. Ein Repositorium ist hierbei die Verknüpfung zwischen der Entwicklung von Komponenten mit dem Ziel der Wiederverwendung und der Entwicklung auf Basis wieder verwendbarer Komponenten. Idealerweise stehen einem Entwickler viele Komponenten zur Verfügung. Durch die Vielzahl an Informationen unterschiedlichster Art je Komponente wäre ein Entwickler bei der manuellen Auswahl von Komponenten überfordert. Daher können Komponenten-Repositorien eingesetzt werden, um eine systematische Informationsauswertung und somit eine Unterstützung des Auswahlprozesses geeigneter Komponenten zu ermöglichen. Die Chance des Entwicklers eine fertige Komponente zu verwenden, anstatt eine neue Komponente zu entwickeln, hängt stark von den Suchfunktionalitäten eines Komponenten-Repositoriums, der Fähigkeit eines Entwicklers die angebotenen Suchfunktionen sinnvoll einzusetzen und der Verfügbarkeit geeigneter Komponenten ab.

4.4.2.1 Komponenten-Beschreibung

Ohne eine strukturierte Beschreibung lässt sich kein Komponenten-Repositorium aufbauen, denn sonst müsste man das Repositorium „von vorne bis hinten“ durchblättern (manuelle Suche) und jede Beschreibung einer Komponente vollständig durchlesen. Da dies bei vielen tausend Software-Artefakten – abgesehen von der nicht mehr durchführbaren maschinellen Vorauswahl von Komponenten – schwer möglich ist, muss der Beschreibung der Komponenten eine Struktur gegeben werden, der sich der Katalog anpassen kann und die auch vom Anwender verstanden wird.

---

[574] Vgl. Sametinger 1997, S. 178.

Sametinger teilt die Komponenten in die vier Basiskategorien Progammschnittstelle, Benutzungsschnittstelle, Datenschnittstelle und Plattform ein, auf der die Komponente abläuft (dazu gehören auch notwendige Programmiersprachen, Hardware, Bibliotheken etc.)[575]. Diese Einteilung berücksichtigt – neben vielen anderen[576] – nicht den eigentlichen Zweck, für den Komponenten geschaffen werden, denn sie ist stark technologieorientiert. Dieses Problem wurde spätestens mit den ersten Ansätzen, Komponenten unternehmensintern für verschiedene Entwickler- und Anwendergruppen zur Verfügung zu stellen, erkannt.

Die Gesellschaft für Informatik e.V. (GI) hat deswegen einen Vorschlag zur Komponentenspezifikation (siehe Abschnitt 3.5.2.6) veröffentlicht, die die möglichst vollständige, eindeutige und widerspruchsfreie Beschreibung der Außenansicht einer Komponente abdecken soll – darunter fällt auch die Beschreibung der Anwendungsdomäne einer Fachkomponente bzw. die generische Funktion einer Systemkomponente. Auch wenn dieser Vorschlag den B2B-Bereich mit Fachanwendern fokussiert, bildet er doch durch seine Erweiterungsfähigkeit bspw. für „Konsumentenkomponenten" einen brauchbaren Ansatz zur Komponentenbeschreibung im B2C-Umfeld unter Mobilitätsaspekten.

Eine Komponentenspezifikation sollte möglichst umfassend, muss aber nicht vollständig sein. Kritisch zu dem Spezifikationsrahmen der GI äußern sich Fettke, Loos und von der Tann[577], die bemängeln, dass es Abhängigkeiten zwischen den Beschreibungsebenen gibt, die den Einsatz von Werkzeugen erfordern, damit die benutzten Begriffe über alle Ebenen konsistent gehalten werden können. Ein weiterer Kritikpunkt ihrerseits ist die Möglichkeit, externe Beschreibungen von Komponentenmerkmalen als Teil der Spezifikation zu integrieren. Auch dies werde zu Widersprüchen und Inkonsistenzen führen. Deshalb hat auch diese Beschreibung auf Basis einer genormten Terminologie zu erfolgen.

Im Folgenden werden Anpassungen des GI-Spezifikationsrahmens an das B2C-Umfeld und mobile Endgeräte vorgeschlagen. Als wichtigste Anpassung muss im Vermarktungsaspekt die Domänenbeschreibung, also der Anwendungsbereich, an die Umwelt des Konsumenten angepasst werden. Als Domäne können bspw. die in Ab-

[575] Vgl. Sametinger 1997, S. 117f.
[576] Vgl. Sametinger 1997, S. 122f.
[577] Vgl. Fettke; Loos; von der Tann 2002.

schnitt 3.3.3 von Ortner hergeleiteten Anwendungsfelder von Menschen in ihrem Alltag (Unterhaltung, Bildung, Gesundheit, unmittelbare Lebensaktivitäten, Organisation, Arbeit und Finanzen) herangezogen werden. Eine weitere Einteilungsmöglichkeit ergibt sich durch eine Klassifikation von Endgeräten nach dem Verwendungszweck (siehe Abschnitt 3.5.1.1). Diese Klassifikation könnte auch für die Komponenteneinteilung herangezogen werden, da diese auch einen Zweck erfüllen. So dienen Spiele für Mobiltelefone bspw. dem Zweck der Unterhaltung. Komponenten, die Verbindungen zu Datenbanken im Internet herstellen oder aktuelle Nachrichten liefern, dienen dem Informationszugang.

Je nach verwendeter Klassifikation werden viele Komponenten in mehrere Kategorien fallen, und es werden Komponenten zu finden sein, die in keine Kategorie gehören – eine vollkommene Einteilung ist in diesem Fall noch nicht absehbar. Doch bietet eine solche Domänenklassifikation dem Anwender einen groben Überblick über am Marktplatz vorhandenen Themengebiete und einen Einstiegspunkt zum Suchen oder Browsen (dt. Stöbern). Auch im Konzeptionsteil muss die Aufgabe der Komponente (ihr thematischer Inhalt), die eine grobe Domänenangabe eingehender beschreibt, an mögliche Aufgaben im Alltag angepasst werden.

Der Aufbau einer einheitlichen Terminologie zwischen Entwickler und Fachanwender ist im B2B-Umfeld bereits eine Notwendigkeit, um für die Anwendungsentwicklung eine gemeinsame Begriffsbasis zu verwenden und bei der späteren Benutzung *„keine Überraschungen zu erleben*[578]*“*. Auch wenn eine solche Terminologie für den privaten Bereich schwer zu erstellen ist, müssen doch zumindest die in der Komponentenspezifikation verwendeten Begriffe in einem „Konsumentenlexikon“ genau und eindeutig in der Gebrauchssprache der Konsumenten definiert werden.

Der Technologieteil muss die erweiterten Möglichkeiten der mobilen Endgeräte berücksichtigen. Hier ist z. B. an eine Erweiterung der Abstimmungsspezifikation zu denken. Neben den zeitlichen Abstimmungen mit anderen Komponenten kann die räumliche Dimension mit einbezogen werden. Damit kann bspw. eine Komponente genau dann in das Anwendungssystem integriert werden, wenn ein bestimmter Aufenthaltsort erreicht wurde.

[578] Overhage 2002, S. 8.

Die eingesetzten Sprachen zur Beschreibung der technologischen Sachverhalte müssen nicht an das Sprachniveau der Nutzer angepasst werden. Es ist von vornherein auszuschließen, dass sich alle Anwender mit der technologischen Konfiguration – und damit auch den Schnittstellen – beschäftigen. Dies ist Aufgabe einer automatischen (Vor-)Auswahl, die das mobile Endgerät und das Marktplatzsystem für den Anwender selbstständig vornehmen. Ein Anwender soll davon ausgehen können, dass alle angezeigten Komponenten nahtlos in sein Anwendungssystem integrierbar sind.

4.4.2.2 Komponenten-Suche

Zur Unterstützung der Suche nach geeigneten Komponenten können Klassifikationsansätze, wie sie bspw. Sametinger[579] aufzählt, verwendet werden. Griffel[580] schlägt eine Repräsentation der Semantik einer Komponente als Beschreibungsrahmen vor. Milli et al.[581] unterteilen Beschreibungssprachen grob in die Kategorien textbasierte, lexikalische und spezifikationsbasierte Beschreibungen. Dabei stellen Milli et al. fest, dass eine spezifikationsbasierte Beschreibung dem Ziel einer Äquivalenz zwischen dem, was eine Komponente an Aufgaben verrichtet und wie sie kodiert (implementiert) wurde, am nächsten kommt.

Effiziente Auswahl von (Software-)Komponenten

Der Erfolg eines Marktplatzsystems für den Komponenten-Handel hängt eng mit der Komponentensuche zusammen, denn die Auswahl der richtigen Komponente ist trotz vielfältiger technischer Fortschritte weiterhin problematisch[582]. Ein Marktplatz, der zwar zahlreiche Komponenten anbietet, dessen Suchsystem jedoch mangelhaft arbeitet und keine oder falsche Komponenten in der Antwort auf die Suchanfrage zurück gibt, kann keinen ökonomischen Erfolg haben. Eine „gute“ Suche zu realisieren ist somit die zentrale Aufgabe eines Marktplatzsystems. Gut bedeutet dabei, dass das System nur inhaltlich zur Anfrage passende Komponenten anbietet und diese technologisch in das Endgerät des Anwenders integrierbar sind. Um geeignete Komponenten effizient suchen zu können, muss das System die Komponenten möglichst vollständig und anwenderfreundlich beschreiben sowie die Benutzung der natürlichen Sprache unterstützen (selbstständige Transformation von natürlicher Sprache). Hierbei ist der Einsatz von Filtern hilfreich, die eine Auswahl der Komponenten sinnvoll einschränken und angepasste Suchstrategien zur Verfügung stellen.

---

[579] Vgl. Sametinger 1997, S. 179-184.
[580] Vgl. Griffel 1998, S. 19 und 70-75.
[581] Vgl. Mili; Mili; Mili 1995, S. 551.
[582] Vgl. Ortner; Overhage 2003b, S. 29.

Transformation umgangssprachlicher Anfragen

Bei der Anwendungsentwicklung ist darauf zu achten, dass Anwender meist nicht in der Lage sind, ihr implizites Wissen in explizites Wissen, d. h. in ein (Handlungs-)Schema, zu transformieren[583]. Dadurch besteht die Gefahr, dass die Softwareingenieure an den Anwendern vorbei Programme entwickeln, die nicht deren Erfordernissen entsprechen. Das Problem wird im Rahmen des Software-Entwicklungsprozesses durch Vorstudien mit Fachgesprächen, dem Aufbau einer Wissensbank (in Analogie zu einer Datenbank), Lexika mit Begriffsdefinitionen usw. angegangen.

Für das Marktplatzsystem ist dieses Problem auf die Suche geeigneter (Software-)Komponenten übertragbar. Es stellt sich hier aber noch schwieriger zu lösen dar, da eine Lösung ad hoc gefunden werden muss. Denn es steht weder das Wissen der Fachanwender als Modell oder Schema zur Verfügung (es existieren keine Fachanwender) noch sind die Kunden einer bestimmten Personengruppe eines genau definierbaren Umfeldes, wie z. B. einer Abteilung eines Unternehmens zuzuordnen.

Dies bedeutet für das Suchsystem als eine Kernkomponente des Marktplatzsystems, dass es die Benutzung der Gebrauchssprache (natürliche Sprache) unterstützen und selbstständig die Transformation in technologische Sprachen, wie bspw. die Datenbanksprache Structured Query Language (SQL) vornehmen muss.

Auch wenn dieses Problem prinzipiell ohne funktionierende künstliche Intelligenz, die vollständig den Sinn bzw. die Semantik jeder menschlichen Aussage versteht, nicht zu lösen sein wird, gibt es doch eine Reihe von Hilfsmitteln die eine solche Transformation unterstützen. Diese Hilfsmittel werden vom Ansatz her im Folgenden kurz vorgestellt.

Die in die Referenzarchitektur (siehe Abschnitt 4.5) integrierten Schlussfolgerungs- und Erwägungssysteme können sprachliche Heterogenitäten nicht nur zwischen menschlichen Kommunikationspartnern meistern. Vielmehr kann ihr Einsatz auch auf die Benutzung von umgangssprachlichen Sätzen in der Eingabemaske einer Suchkomponente erweitert werden.

Der Anwender kann selbst aktiv werden und das Begriffslexikon verwenden, um passende Suchbegriffe für seine Anfrage zu finden. Das Begriffslexikon muss ständig an

[583] Vgl. Elzenheimer; Lonthoff; Ortner 2006, S. 580.

die Gebrauchssprache angepasst werden, um auch moderne (aktuelle) Begriffe wie z. B. „Handy“ zu enthalten.

Ein weiteres Hilfsmittel bietet die Sprachtheorie mit der Beseitigung von Sprachdefekten (siehe Abschnitt 2.5.1.1). Sprachdefekte sind unscharfe, falsche oder widersprüchliche Begriffe, deren Beseitigung ursprünglich zum Aufbau einer unternehmensweiten Aussagensammlung in der Softwareentwicklung diente[584]. Sie können aber auch im Suchsystem eingesetzt werden, um klare, einheitliche Begriffe zu vereinbaren und somit die Sprache zu präzisieren.

Eine Analyse der gesuchten Begriffe und weitere statistische Auswertungen, wie bspw. mit Hilfe der formalen Begriffsanalyse[585], können helfen, dem Anwender zumindest das anzubieten, was Anwender vor ihm mit ähnlichen Suchbegriffen letztlich an Komponenten ausgewählt haben.

Vielfach wird in wissenschaftlicher Literatur[586] gefordert, Anwendern eine Fachsprache bezüglich der Computernutzung näher zu bringen. Konsequent angewendet würde eine solche Maßnahme zum Verschwinden sprachlicher Vielfalt und im Extremfall zur Reduktion der Sprache auf maschinenverständliche Befehlswörter führen. Es bleibt aber dagegen eine menschliche Stärke, aus Erfahrung zu lernen. Schon heute benutzen viele Anwender Internet-Suchmaschinen ganz selbstverständlich. Sie haben gelernt, die richtigen Suchwörter einzugeben, um passende Seiten zu finden. Die Hoffnung, dass sich diese Erfahrung auch auf die Komponentensuche niederschlägt, ist sicher nicht unbegründet. Damit kann der Anwender von selbst durch Übung zum „Komponenten-Fachanwender“ werden[587].

Filter zur automatisierten Vorauswahl

Neben der Sprachanpassung gibt es eine Reihe von Möglichkeiten auf technischer Ebene, die Suche nach (Software-)Komponenten effizient zu gestalten. Eine automatische Vorauswahl von Komponenten reduziert die Menge an Komponenten, aus denen der Anwender sich die passende aussuchen muss. Komponenten im Katalog oder eine Suchantwort aufzulisten, die technisch nicht auf mindestens einem Endgerät des Nutzers funktionsfähig sind, ist sinnlos. Komponenten, die bereits auf dem Endgerät in-

[584] Vgl. Ortner 2005a, S. 68f.
[585] Vgl. Wille; Ganter 1996.
[586] Vgl. Heinemann 2006, Kapitel 7, S. 64ff.
[587] Vgl. Babiak 1999, S. 17.

stalliert sind, brauchen auch nicht noch einmal in den Fokus des Benutzers zu geraten, mit Ausnahme von Aktualisierungen vorhandener Komponenten.

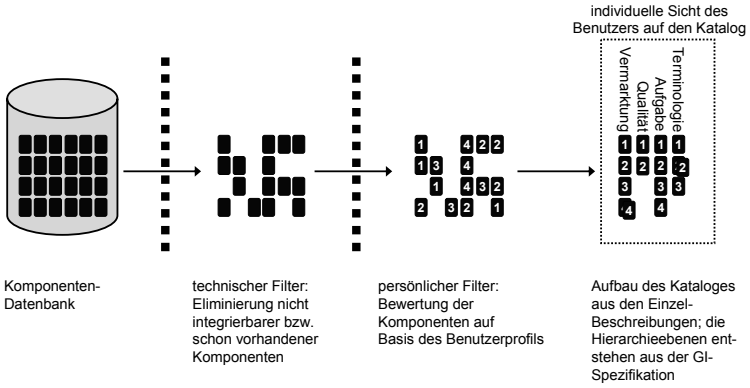

Abbildung 53: Schematischer Ablauf der Katalogentstehung unter Einsatz von technischen und persönlichen Präferenzen

Dieser technische Filter kann durch einen die persönlichen Präferenzen auswertenden Filter ergänzt werden. Der letztgenannte Filter muss nicht zwangsläufig „unerwünschte" Komponenten aus der Ergebnisliste ausblenden, kann aber mit einer geringeren Bewertung solcher (Software-)Komponenten für den Benutzer eine Entscheidungshilfe sein, da sie in der Ergebnisliste und im Katalog an unteren Stellen erscheinen (Relevanzsortierung). Abbildung 53 verdeutlicht die Art und Reihenfolge der benutzten Filter. Diese Informationen bezieht das Filtersystem aus dem Benutzerprofil (siehe Abschnitt 4.4.3).

4.4.3 Benutzerprofile

Sämtliche Eigenschaften eines Anwenders lassen sich in einem Benutzerprofil bündeln, was immer dann sinnvoll ist, wenn man nicht überall sämtliche Parameter selbst eingeben möchte oder durch Vorselektion ein personalisiertes Marktplatzangebot dem Kunden anbieten möchte. Benutzerprofile sind Informationen über Eigenschaften, Präferenzen oder Verhaltensweisen eines Anwenders[588]. Auch die Ausstattung des Anwenders mit den ihm zur Verfügung stehenden mobilen Endgeräten kann mit einbezogen werden.

Es gibt verschiedene Einteilungsmöglichkeiten für Benutzerprofile. Die unmittelbarste ist die Einteilung nach dem Beteiligungsgrad des Anwenders an ihrer Entstehung. Da-

[588] Vgl. Merz 2002, S. 517.

durch lassen sie sich in explizite, abgeleitete und verdichtete Daten unterteilen[589]. Explizite Daten sind vom Anwender bewusst bereitgestellte Daten, die in das Profil einfließen, wie bspw.

- persönliche Daten, wie Adressen, Bankverbindungen oder Alter,
- Präferenzen, wie Hobbys oder Lieblingstiere,
- Sicherheitsprofile, wie die gewünschte Verschlüsselung der Kommunikation und
- Kommunikationsparameter wie die vorhandene Bandbreite.

Abgeleiteten Daten stammen nur indirekt vom Anwender, sie werden meist ohne sein Wissen im Hintergrund gesammelt. Zu diesen Daten gehören bspw. die Sitzungsdauer, alle Informationen im Zusammenhang mit getätigten Käufen (Transaktionsdaten), Produktkombinationen im Warenkorb und die vom Anwender verwendeten Suchbegriffe. Weitere für Marketingzwecke interessante Daten sind Seitenzugriffe eines Anwenders auf einer Webseite (engl. Hits), der Pfad durch die Webseiten (engl. Clickstreams) und Klicks auf Werbebanner.

Verdichtete Daten stammen aus den ersten beiden Kategorien und werden häufig in Data Warehouses gesammelt. Mit Hilfe von Online Analytical Processing-Werkzeugen (OLAP) und Data Mining werden sie aggregiert und anschließend analysiert. Solche Daten sind keinem Benutzer mehr zuordenbar. Sie dienen dem Marktplatz-Betreiber zu Marketingzwecken, der Marktforschung und ggf. der Verbesserung des Qualitätsstandards bspw. der vom Marktplatzsystem bereitgestellten Webseiten.

Benutzerdaten können auch hinsichtlich ihrer Lebensdauer in statische oder dynamische Daten eingeteilt werden. Statische Daten sind bspw. die Anschrift des Anwenders (gegenüber dem dynamischen Aufenthaltsort), seine Bankverbindung und die vorhandenen mobilen Endgeräte. Neben der aktuellen Position des Anwenders sind dessen Bereitschaft, zu kommunizieren, das aktive Endgerät etc. dynamische Parameter.

Für einen „mobilen Marktplatz“ ist eine Einteilung nach Herkunft der Daten sinnvoll, so wie sie Merz vorschlägt. Diese Einteilung lässt sich den Objekten Benutzer, mobiles Endgerät und Marktplatz zuordnen. Die erhobenen Daten werden – nachdem sie in den Datenbanken bzw. im Repositorium zur Verfügung stehen – zu verschiedenen Zwecken benutzt, die im Folgenden aufgeführt werden:

[589] Vgl. Merz 2002, S. 523.

- *Ausnutzung des Automatisierungspotenzials bezüglich der Dateneingabe*
  Durch die Speicherung von Benutzerdaten müssen langwierige Registrierungseingaben, die auf den meisten Marktplätzen anfallen, nur bei der ersten Anmeldung erfolgen (Single SignOn). Dies ist für den Anwender schon auf PCs eine Erleichterung, bei mobilen Endgeräten mit ihren meist unzulänglichen Tastaturen jedoch noch effektiver. Die Schreibarbeit wird durch das Vermeiden von redundanter Informationseingabe wesentlich reduziert. Auch beim Marktplatzsystem erhöht sich der Grad an interner Automatisierung, da die eingegebenen Daten jederzeit über die Datenbanken abrufbar sind.

- *Layoutanpassung an persönliche Präferenzen und Erfordernisse*
  Durch das Wissen um die Präferenzen bei der Darstellung lässt sich die Ausgabe personalisieren. Für behinderte Menschen kann diese Anpassung (z. B. das Einstellen einer großen Schrift) zur Lesbarkeit von Text notwendig werden.

- *Filterfunktionen für die Suche und Angebotserstellung*
  Auf Grund der Kenntnis von Geräteeinstellungen und Benutzerpräferenzen kann die Suche gefiltert werden und der Katalog dynamisch individualisiert werden. Außerdem gelingt es mit solchen Daten, dem Anwender aktiv Angebote zukommen zu lassen, die seine spezifischen Bedürfnisse genau treffen. Dies kann bei geeigneter Umsetzung zum echten Mehrwert werden und damit die Kundenbindung steigern.

- *Automatische (Software-)Komponentenanpassung*
  Nachdem die Suche nach Komponenten bereits durch Benutzerprofile unterstützt wird, bietet sich deren Hilfe auch bei der Anpassung von Komponenten an. Hier hilft ein Profil bei der Auswahl einer Komponente (die in Variantenfertigung in großer Stückzahl vorliegt), bei Parameteranpassungen von im Quelltext vorliegenden Komponenten (Modifikation) und der anschließenden Zusammensetzung von Komponenten zu Baugruppen und Anwendungen (Konfiguration).

- *Transferverbesserungen*
  Durch Kenntnis von Bewegungsmustern, Aufenthaltsorten mit jeweiliger Verbindungsqualität (z. B. Bandbreite) sowie Verbindungsparametern lässt sich der Datentransfer positiv beeinflussen.

- *Benutzerdaten als ökonomischer Faktor*
  Über Kunden gewonnene Daten lassen sich nicht nur für eigene Zwecke gebrauchen, sondern auch als Gut – *„Kundenprofile sind ein wertvoller Schatz*[590]*“* – handeln. Dabei ist zwischen persönlichen Daten mit Namen und Anschrift des Kunden und verdichteten und somit anonymisierten Informationen zu unterscheiden. Während die erste Art von Daten datenschutzrechtlich bedenklich ist und zumindest die explizite Zustimmung des Kunden zur Sammlung und Weitergabe erfordert, können die verdichteten Daten meist problemlos kommerziell ausgewertet werden.

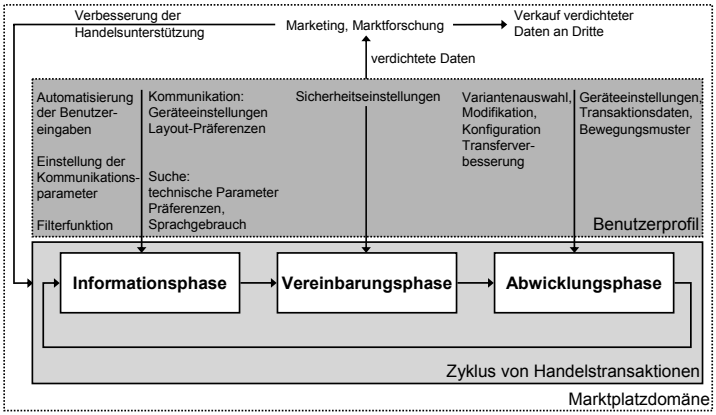

Abbildung 54: Einfluss des Benutzerprofils auf den Marktplatz

Letztlich dienen die Informationen dazu, Kosten zu reduzieren (Automatisierung) und die Qualität der Anwendung sowie der angebotenen Komponenten zu verbessern. Diese Qualitätsverbesserung führt in einer Kausalkette von Kundenzufriedenheit über Kundenbindung hin zu langfristigem ökonomischem Erfolg[591]. Abbildung 54 fasst die in diesem Abschnitt beschriebenen Beziehungen zwischen Benutzerprofil, dessen Einfluss auf die Handelsphasen und die erreichbaren Qualitätsverbesserungen zusammen.

4.4.4 Geräteprofile

Die Problematik mobiler Endgeräte besteht in der ausgeprägten Heterogenität der unterschiedlichen Endgeräte. Eine Übersicht im Rahmen von Klassifikationsversuchen wurde in Abschnitt 3.5.1.1 dargestellt. Damit das Marktplatzsystem diese Heterogenität verwalten kann, werden aus Geräteklassen Geräteprofile abgeleitet. Damit der Be-

[590] Zobel 2001, S. 231.
[591] Vgl. Meffert 2000, S. 366 und 947.

nutzer des mobilen Endgerätes, die für sein Endgerät optimal verfügbare Darstellung der Inhalte des Marktplatzes erhält, werden dem elektronischen Marktplatz schon bei der Informationsphase entsprechende Geräteprofildaten bereitgestellt. Somit kann der Zugang zu dem Marktplatzsystem gewährleistet werden, indem bspw. zur Darstellung die vorhandene Bildschirmauflösung berücksichtigt wird. Zur Unterstützung der Suche nach geeigneten (Software-)Komponenten für ein mobiles Endgerät werden Informationen zu den Geräteressourcen sowie Betriebssystemversionen und Laufzeitumgebungen aus dem Geräteprofil benötigt.

Eine (Software-)Komponente, die auf dem speziellen Endgerät nicht betriebsfähig ist, interessiert den Anwender nicht. Somit kann schon aufgrund des Geräteprofils das Angebot des Marktplatzes vorgefiltert werden. Darüber hinaus sollte dennoch eine manuelle Suche nach (Software-)Komponenten möglich sein, die für andere Endgeräte geeignet ist, falls das Marktzugangsgerät nicht gleich dem gewünschten Zielgerät entspricht.

4.4.5 Kommunikationsart

Mobile Kommunikation ist dadurch gekennzeichnet, dass sie auf unzuverlässige Netzwerkressourcen zugreift. Mobile Kommunikation kann dabei sowohl verbindungsorientiert (GSM-Verbindung) als auch verbindungslos (GPRS über GSM-Netze bzw. UMTS-Netz) erfolgen. Die verfügbare Bandbreite ist variabel und je nach Infrastruktur für mobile Datenübertragung ausreichend oder sehr knapp. Hinzu kommt die Fehleranfälligkeit der Kommunikation, die fehlertolerante Übertragungsmechanismen erfordert.

Die geforderte Mobilität der Anwender und seiner Endgeräte soll auch während der Kommunikation mit dem Marktplatz aufrecht erhalten bleiben. Hier stellt sich die Frage, welche Systeme und Protokolle der drahtlosen Übertragungstechnologien (siehe Abschnitt 3.5.1.2) geeignete Kommunikationsmöglichkeiten bieten.

Das Netzwerk sollte folgende Eigenschaften besitzen: hohe (Flächen-)Abdeckung, Unabhängigkeit von der Geschwindigkeit des Anwenders, transparenter Verbindungswechsel, bidirektionale Kommunikation und die Möglichkeit einer zweckmäßigen Positionsbestimmung.

Die erste Anforderung, eine hohe Netzwerkabdeckung, schließt Funksysteme mit kurzer Reichweite, wie z. B. Bluetooth oder WLAN aus. Die sogenannten Long-Range-Netzwerke werden vor allem durch das GSM- und UMTS-Mobilfunknetz repräsentiert. Diese Netzwerke erfüllen auch die anderen Punkte der Kriterienliste. Für eine paketorientierte Datenübertragung zwischen Marktplatzsystem und mobilem Endgerät bietet sich das auch von der Bandbreite geeignete GPRS über GSM oder UMTS an.

4.4.6 Fehlertoleranter Dateitransfer

Da die Kommunikation in drahtlosen Netzwerken im Allgemeinen fehleranfällig ist, muss das Marktplatzsystem für einen robusten Dateitransfer sorgen.

Auf Grundlage der in Abschnitt 3.5.1.2 vorgestellten Kommunikationsnetze lassen sich verschiedene Verfahren realisieren, die für einen korrekten Datentransfer sorgen. Dabei ist grundsätzlich zwischen Pull- und Push-Technologie zu unterscheiden. Während bei der Pull-Technologie der Anwender oder das mobile Endgerät, also der Client, den Dateitransfer initiiert, wird beim „pushen“ der Transfer vom Server automatisch eingeleitet[592]. Unabhängig von der eingesetzten Technologie sollten die Transferapplikationen die von Häckelmann et al.[593] benannten Eigenschaften erfüllen, die – für mobile Endgeräte leicht modifiziert – hier wiedergegeben werden:

- *Beherrschung heterogener Umgebung*
  Im mobilen Umfeld muss mit unterschiedlichen Kommunikationstechnologien und unterschiedlichen Standards gerechnet werden. Da zwischen Server (dem Marktplatzsystem) und Client (dem mobilen Endgerät) auch unterschiedliche Betriebssysteme vorliegen, muss ein Transfersystem Dateien konvertieren können (Typ- und Formatanpassungen).

- *Lesen und Schreiben von Dateien, Verzeichnissen und Dateiattributen*
  Neben dem eigentlichen Bereitstellen (z. B. durch Herunterladen) der Datei kann es notwendig sein, dass das System eigene Verzeichnisse auf dem mobilen Endgerät anlegt. Die transferierte Datei kann auch mit einem Schreibschutz versehen werden, damit sie nicht versehentlich vom Anwender gelöscht wird. Nach dem Herunterladen sollte das System überprüfen können, ob der Dateitransfer erfolgreich war. Dazu liest das System die übertragene Datei ein weiteres Mal und ver-

[592] Vgl. Lehner 2003, S. 148f.
[593] Vgl. Häckelmann; Petzold; Strahringer 2000, S. 407.

gleicht die einzelnen Bits mit dem Original oder benutzt die effizientere Methode der Prüfsummenkorrektur.

- *Unterstützung verschiedener Dateitypen und -formate*
  Eine Formatanpassung oder Typkonvertierung kann notwendig werden, um verschiedene Dateitypen bzw. -formate zu unterstützen. Diese Forderung hängt eng mit der Beherrschung heterogener Umgebungen zusammen.

- *Unterstützung eines partiellen Dateizugriffs*
  Falls Teile der Datei falsch oder gar nicht übertragen wurden, kann eine Neuübertragung ggf. nur bestimmte Teile der Datei betreffen. Idealerweise setzt die erneute Übertragung an der Stelle ein, an der die Unterbrechung auftrat (Resuming). Dadurch müssen nicht noch einmal umfangreiche Daten kostspielig und redundant gesendet werden.

- *Unterstützung verschiedener Client-seitiger Schnittstellen*
  Neben der Benutzungsschnittstelle sollte der Client auch die Ausführung durch Drittprogramme unterstützen, so dass z. B. das mobile Endgerät ohne Anwendereingriff automatisch neue Komponenten herunterladen kann.

Gerade bei Übertragungen zu mobilen Endgeräten wird der partielle Dateizugriff sehr bedeutend, da es trotz des transparenten Übergangs vom Internet zu mobilem Endgerät im Funkbereich immer wieder zu Verbindungsabbrüchen und Bitfehlern kommen wird. Grund dafür ist, dass sich ein Funkkanal nicht wie ein Kabel gegen Störeinflüsse abschirmen lässt. Während sich einzelne Bitfehler lediglich in der Geschwindigkeit auswirken und durch das Protokoll automatisch erkannt und korrigiert werden, können Verbindungsabbrüche auf Grund von Einfahrten in Tunnel, leeren Energiespeichern, Hardwaredefekten, Netzwerküberlastung etc. nicht vermieden werden. Das Konzept des Resumings setzt hier an, wie bspw. beim File Transfer Protocol (FTP) oder File Transfer, Access and Management (FTAM). Da die GPRS-Technik die vollständig transparente Unterstützung des Internet-Protokolls TCP/IP gewährleistet, kann ein FTP-Client auf dem GPRS-gestützten mobilen Endgerät und ein FTP-Server im Marktplatzsystem das Resuming ermöglichen. Ebenso wird im WAP mit Hilfe des

Wireless Transaction Protocol (WTP) ein zuverlässiger Nachrichtenaustausch über unzuverlässige Netzwerke realisiert[594].

Benötigt wird ein netzwerkfähiges, für mobile Endgeräte einsetzbares Dateisystem mit Journaling-Fähigkeiten[595]. Der Server kopiert die Datei, die die (Software-)Komponente enthält, auf das mobile Endgerät, indem er z. B. die IP-Adresse als virtuelles Verzeichnis benutzt (das Netzwerkprotokoll muss dazu wie beim FTP-Transfer wiederum TCP/IP unterstützen). Durch das Prinzip des Journaling wird eine vollständige und konsistente Übertragung gewährleistet.

4.4.7 Lokalisierung

Durch die mobile Verfügbarkeit von Anwendungen und durch die Nutzung mobiler Endgeräte erhält der mobile Marktplatz einen Mehrwert durch Verwendung der Ortsinformation. Der Aufenthaltsort eines Anwenders kann durch Lokalisierung des mobilen Endgerätes ermittelt werden. Eine Lokalisierung kann dabei durch den Einsatz unterschiedlicher Lokalisierungsmethoden erfolgen. Hierbei können zunächst netzwerkbasierte und terminalbasierte Methoden unterschieden werden[596]. Weiterhin gibt es satellitengestützte Lokalisierungsmethoden und Hybridansätze.

Netwerkbasierte Methoden

Bei netzwerkbasierten Methoden zur Lokalisierung übernimmt das Netzwerk die Aufgabe der Positionsbestimmung. Auf der Basis der GSM-Netze (gilt analog für UMTS) gibt es im Wesentlichen folgende verschiedene Methoden der Lokalisierung, die hier ansatzweise vorgestellt werden[597]:

Cell of Origin (COO) ist die einfachste Lokalisierungsmethode. Hierbei wird die Funkzelle (engl. Cell) bzw. die Basisstation, in der sich ein Anwender aufhält, identifiziert. Diese muss zur Kommunikation zwangsläufig bekannt sein. Da das Mobiltelefon regelmäßig Kontakt mit der Funkzelle aufnimmt, ist der Aufenthaltsort bei eingeschaltetem Endgerät bis auf einen Radius von ca. 100 m in städtischen Gebieten mit dichter Funkzellenbesiedlung bestimmbar, in ländlichen Gebieten erweitert sich der Radius

[594] Vgl. Roth 2005, S. 248.
[595] Journaling bezeichnet das Aufzeichnen aller Änderungen bis zum erfolgreichen Abschluss einer Transaktion, wie z. B. einem Schreibvorgang in einem Dateisystem.
[596] Vgl. Lonthoff; Ortner 2006b, S. 486f.
[597] Vgl. Röttger-Gerigk 2002, S. 419ff.

dagegen stark (bis zu 35 km). Die Positionsgenauigkeit hängt hierbei direkt von der Größe einer Funkzelle ab.

Angle of Arrival (AOA) benutzt zur Lokalisierung zwei Funkzellen, die jeweils die Richtung des eintreffenden Funksignals des Mobiltelefons bestimmen. Durch diese „Kreuzpeilung“ lässt sich die Position des Endgerätes auf ca. 120 m genau bestimmen, vor allem dann, wenn sich der mobile Teilnehmer nahe bei den Basisstationen befindet. Ein Problem hierbei stellt die Reflexion der Funkwellen an Bergen und Gebäuden dar.

Time Difference of Arrival (TDOA) konstruiert aus den Laufzeitdifferenzen von Funksignalen zweier Basisstationen eine Hyperbel um eine Basisstation. Auf dieser Hyperbel muss sich das mobile Endgerät folglich aufhalten. Durch den Einbezug einer dritten Basisstation lässt sich dann der Aufenthaltsort (Positionsgenauigkeit ca. 50 m) bestimmen.

Time of Arrival (TOA) bestimmt statt der Differenz der Laufzeit die tatsächliche Laufzeit von Funksignalen. Bei zwei Basisstationen, die die Signale des Mobiltelefons empfangen, lässt sich damit die Position des mobilen Endgeräts auf zwei mögliche Punkte genau bestimmen. Durch den Einbezug einer dritten Basisstation wird der Aufenthaltsort eindeutig. In der Regel wird hierbei eine Positionsgenauigkeit von ca. 150 m erreicht.

RadioCamera System nutzt die Eigenschaft aus, dass jede von einem bestimmten Ort ausgesendete Funkwelle ein spezifisches Muster aufweist. Zur Lokalisierung wird das Muster von der empfangenden Basisstation mit den gespeicherten Mustern aus einer Datenbank abgeglichen (engl. Location Pattern Matching). Weil die Entstehungsorte der abgespeicherten Muster bekannt sind, kann auch für das neue Signal die Herkunft bestimmt werden. Damit eignet sich das Verfahren besonders in Städten, da andere Verfahren dort Probleme aufgrund von Reflexionen hoher Wände haben.

Signal Attenuation (SA) misst statt der Laufzeit der Funksignale die Signalstärke. Da die Signalstärke durch Hindernisse und Witterung stark schwankt, ist diese Lokalisierungsmethode sehr ungenau und liefert daher eine Positionsgenauigkeit von bestenfalls 300 m.

Terminalbasierte Methoden

Terminalbasierte Methoden ermitteln die Position aufgrund bereitstehender Informationen auf dem mobilen Endgerät.

Cell of Origin ist auch als ein terminalbasiertes Verfahren anzusehen, da die jeweilige Cell-Id direkt aus dem mobilen Endgerät (Terminal) ausgelesen werden kann. Hierzu ist allerdings eine Referenzdatenbank notwendig, welche die zu den jeweiligen Cell-Ids gespeicherten Geokoordinaten (Position der zugehörigen Basisstation) enthält.

Enhanced Observed Time Difference (E-OTD) ist eine Lokalisierungsmethode, bei der das mobile Endgerät die Lokalisierung aufgrund der Messung der Zeitdifferenz von Funksignalen, die von drei Basisstationen an das mobile Endgerät gesendet werden durchführt. Durch die Messung der Zeitunterschiede wird eine Positionsgenauigkeit von ca. 50 m erreicht, ähnlich wie bei dem TDOA-Verfahren.

Satellitengestütze Methoden

Satellitentechnologien sind mit dem *Global Positioning System (GPS)* als einem militärischen, amerikanischen, nicht-terrestrischen Lokalisierungssystem vertreten. Es bietet eine recht exakte Positionsbestimmung (etwa zwischen 5 und 50 m) auch in Gebieten ohne jegliche Infrastruktur. *Assisted-GPS (A-GPS)* benutzt – ähnlich wie TOA – die Zeit, um die Distanz zu einem Satelliten zu messen. Sobald die Distanz zu drei Satelliten bekannt ist, kann die exakte Position des Anwenders bestimmt werden. Dieses Verfahren ist auch als Triangulation bekannt. Das System ist ohne Modifikation des Mobilfunknetzes einsatzbereit, allerdings muss das mobile Endgerät über einen GPS-Empfänger verfügen.

Neben GPS gibt es noch *Inmarsat* und das zivile im Aufbau befindliche europäische Satellitennavigationssystem namens *Galileo*, das eine noch präzisere Lokalisierung zulässt (Positionsgenauigkeit ca. 5 m und gegen Gebühr im Zentimeterbereich)[598].

Hybridansatz

Ein weiterer Ansatz, der *Location Trader* des Center for Digital Technology and Management (CDTM) in München, kombiniert Lokalisierungsverfahren und arbeitet ausschließlich mit öffentlich zugänglichen Informationen, wie z. B. Funkzelle und Signalstärke. Mit Hilfe einer Vielzahl von Nutzern (Community) wird iterativ eine Lokalisie-

[598] Vgl. Galileo 2003, S. 6.

rungsdatenbank aufgebaut. Die Datenbasis wird dabei durch explizite Interaktion mit dem Anwender, heuristischen Berechnungen und tabellarischem Suchen und Ergänzen erweitert. Das Ziel ist eine providerunabhängige Lokalisierungsmöglichkeit, die umso präziser wird, je mehr Teilnehmer beteiligt sind[599].

Lokalisierungsnutzung

Durch die Lokalisierungsmöglichkeiten bekommen mobile Anwendungen neben dem zeitlichen Kontext als weiteren Freiheitsgrad einen lokalen Kontext hinzu. Sie erhalten einen Ortsbezug.

Um den Nutzen von Ortsinformationen für Marktplätze evaluieren zu können, wird im Folgenden idealisiert von einem Dienst ausgegangen, der die Information hinreichend genau, netzwerkunabhängig und zeitnah auf einem für Drittanwendungen zugänglichen Server unter Beachtung aller rechtlichen Vorschriften zur Verfügung stellt (bspw. als Web Service). „Hinreichend genau" bedeutet hierbei, dass Anwendungen solche Ortsinformationen zur Verfügung gestellt bekommen, mit denen sie ihre Zwecke erfüllen können. Ein Wetterdienst muss im Gegensatz zu automatisch abgesetzten Notrufen z. B. nur wissen, in welcher Region sich der Anwender befindet – für einen Notarztwagen ist diese Information nicht ausreichend. Der Rettungsdienst muss neben der exakten Ortsangabe auch direkt nach dem Absetzen des Notrufs zeitnah über die Ortsangabe verfügen. Eine solche Architektur wird in José[600] vorgestellt. Die dort beschriebene Architektur liefert neben „harten" Ortsinformationen wie dem Längen- und Breitengrad auch „weiche" bzw. relative Fakten wie „in der Nähe von ..." oder „neben der Bibliothek".

Geht man von diesen (idealisierten) Voraussetzungen aus, kann aus der Lokalisierung von Anwendern in vielfältiger Weise Nutzen gezogen werden. Lokalisierungsnutzung kann zur Kommunikationsverbesserung beitragen, indem rechtzeitig vor Funklöchern gewarnt wird. Lokalisierung ermöglicht eine zusätzliche Dimension der Individualisierbarkeit. Aus dem Bewegungsprofil eines Anwenders können nützliche Informationen abgeleitet werden. Durch die vorhandenen Lokalisierungsdaten können Eingaben des aktuellen Ortes (z. B. für mobile Dienstleister) automatisiert werden. Durch Lokalisierung können ortsbezogene Werbung und ortsbezogene Rabattsysteme angeboten

[599] Vgl. Dornbusch; Huber 2003, S. 175ff.
[600] José; Davis 1999.

werden[601]. Durch Lokalisierung können lokale (ortsabhängige) mobile Spiele ortsabhängig angeboten werden[602]. Lokalisierung kann aber auch als ein separater Zusatzdienst zur Orientierungs- und Navigationshilfe genutzt werden.

4.5 Konzept eines mobilen Marktplatzsystems für den Komponenten-Handel

Aufgrund der Komponentenorientierung und der geforderten Flexibilität orientiert sich das hier zu entwerfende Referenzsystem ausgehend vom Basisschema eines elektronischen Marktplatzes (siehe Abbildung 55) an dem E-NOgS³-Framework (siehe Abschnitt 3.6.1).

Das System muss neben der Unterstützung der Handelsphasen die Aspekte Mobilität und Anwender (i. S. von Usability) besonders berücksichtigen sowie für die Art der gehandelten Produkte, (Software-)Komponenten, geeignet sein.

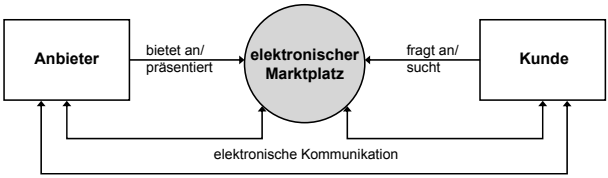

Abbildung 55: Schema eines einfachen elektronischen Marktplatzsystems

In Abbildung 56 wird eine Referenzarchitektur für elektronische Marktplatzsysteme zum Handel von (Software-)Komponenten im mobilen Umfeld dargestellt. Diese Referenzarchitektur baut auf dem E-NOgS³-Architekturmodell (siehe Abschnitt 3.6.1) auf. Die Elemente vom E-NOg-Server sind in Abbildung 56 als ausgefüllte Elemente dargestellt. Die nichtausgefüllten Elemente stellen Komponenten dar, die sehr spezifische Aufgaben übernehmen. Die Referenzarchitektur beschreibt als Erweiterung auch die beteiligten Handelspartner und deren Informationsflüsse sowie die Kommunikation zwischen den Marktplatzkomponenten. Die dargestellte logische, virtuelle Sicht auf das Marktplatzsystem muss nicht der physischen Realisierung entsprechen. Tatsächlich können Komponenten verschmolzen oder in mehrere Teile aufgespalten sein.

[601] Lonthoff; Wolf 2006.
[602] Lonthoff; Ortner 2006b.

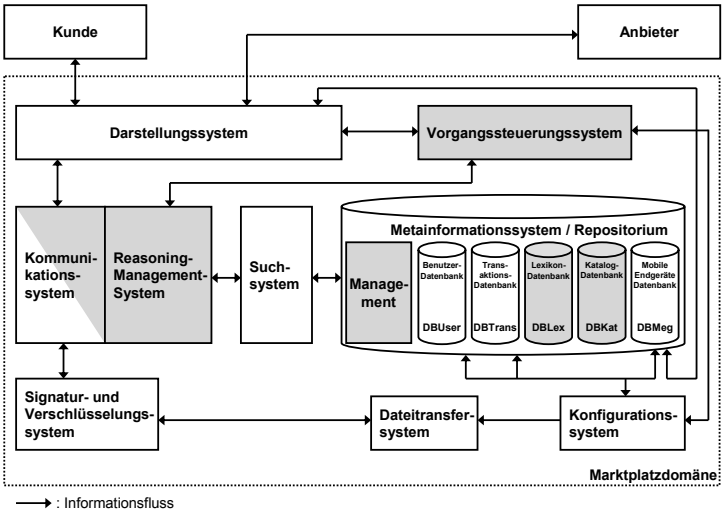

Abbildung 56: Referenzarchitektur für mobile Marktplatzsysteme für den (Software-)Komponenten-Handel auf Basis von E-NOgS³

Nachfolgend werden die Elemente der Referenzarchitektur des Marktplatzsystems und die Beziehungen der Elemente beschrieben.

Darstellungssystem

Das Darstellungssystem bildet den Zugang zum Marktplatzsystem. Es ist daher die zentrale Komponente für Anbieter und Kunden. Je nach Endgerät muss das System eine geeignete Darstellung bzw. Sprache für die Darstellung (oder äquivalente denkbare Mensch-Maschine-Schnittstellen) auf dem mobilen Endgerät finden (Zugriff auf DBMeg).

Ein Mobiltelefon der aktuellen Generation verlangt z. B. die Darstellung der Daten im Format der Wireless Markup Language (WML), die dann mit einem Wireless Application Protocol-Browser (WAP-Browser) abrufbar sind. Ein Notebook kann hingegen mit HTML und HTTP umgehen und ein aktueller PDA wiederum mit beiden Protokollen und Formaten. Durch die Extensible Stylesheet Language for Transformations (XSLT) lassen sich XML-Dokumente einfach in diese und weitere Formate überführen, da Darstellung und Inhalt getrennt abgespeichert werden. Auch die optimale Ausnutzung der durch das mobile Endgerät angebotenen Bildschirmauflösung, kann über XSL-Style Sheets in Verbindung mit der Transformation durch XSLT umgesetzt werden.

Um die für das Endgerät geeignete Sprache zu finden, muss das Darstellungssystem Zugriff auf verschiedene Datenbanken haben, die die Zuordnung von Anwender zu Endgerät und von Endgerät zu Protokoll bzw. Sprache kennen. Im Darstellungssystem wird die syntaktische Heterogenität zwischen Anbieter, Kunde und Marktplatzsystem (Verwendung unterschiedlicher Dokumentformate der Interaktionspartner[603]) überwunden. Die technische Heterogenität (Verwendung unterschiedlicher Kommunikationsnetze und -protokolle) wird mit Hilfe des Kommunikationssystems überwunden.

Falls eine Darstellung für den Anwender nicht notwendig ist, weil sein Endgerät die (Software-)Komponenten automatisiert abruft, kann das Darstellungssystem umgangen werden. Die dann stattfindende Kommunikation zwischen Endgerät und Marktplatzsystem – etwa Suchbefehle oder Parameterübermittlungen zur automatischen Kompilierung der Komponente – wird bevorzugt im darstellungsunabhängigen XML-Format übermittelt. Hierbei lassen sich auch Web Services einsetzen, wie sie im E-NOgS³-System zum Einsatz kommen.

Das Darstellungssystem präsentiert den Marktplatz, die Steuerung übernimmt hingegen das Vorgangssteuerungssystem.

Kommunikationssystem

Das Kommunikationssystem sorgt für eine Vereinheitlichung der von Anbieter und Kunden benutzten Kommunikationsprotokolle, denn durch die vielen verschiedenartigen Endgeräte ist eine Vielzahl an Protokollen zu erwarten. Die Kommunikation kann nach ihrer Normierung bei Bedarf an andere Komponenten, z. B. das Suchsystem, weitergegeben werden. Ein geeignetes Mittel ist hierbei das XML-Format. Die Beschreibung und Strukturierung des Dokumenteninhalts kann damit sinnvoll getrennt werden. Mit Hilfe von DTD oder des umfassenderen XML Schema-Konzepts (XSD) kann zusätzlich eine Grammatik zur Erzeugung gültiger (valider) XML-Dokumente vorgegeben werden.

Die eingehenden Dialogfragmente des Kunden oder des Anbieters werden über das Darstellungssystem vom Kommunikationssystem entgegengenommen, auf das verwendete Protokoll untersucht und bei Bedarf durch einen Wrapper nach XML transformiert. Daraufhin folgt die sofortige Weiterleitung an das Schlussfolgerungs-, Verständigungs- und Erwägungssystem (Reasoning-Management-System), das eng

[603] Vgl. Ortner; Overhage 2003a, S. 25.

mit der Kommunikationskomponente kooperiert. Das System analysiert die Semantik der Kommunikation und die eingesetzten Sprachhandlungstypen (siehe Abschnitt 2.5.1.1). Es kann sich z. B. um eine Suchanfrage nach Komponenten handeln, in welchem Fall die Kommunikation an das Suchsystem weitergeleitet werden muss. Denkbar ist aber auch die Übermittlung der Information neu eingestellter Komponenten im Rahmen von Publish/Subscribe-Mechanismen[604]. In diesem Fall sendet das System die Information an das Kommunikationssystem des potenziellen Kunden. Die korrekte Formatierung ausgehender Kommunikation (wiederum bedingt durch die verschiedenen Protokolle und Sprachen) wird vom Darstellungssystem vorgenommen.

Zur Sicherheit aller Kommunikationsbeteiligten kann die gesamte Kommunikation verschlüsselt und signiert werden. Dies ist Aufgabe des Signatur- und Verschlüsselungssystems.

Reasoning-Management-System

Analog zu dem Schlussfolgerungs-, Verständigungs- und Erwägungssystem aus dem E-NOgS³-System unterstützt diese Komponente die Kommunikation und Interaktion der Teilnehmer, indem es zum einen versucht, durch semantische Heterogenitäten verursachte Kommunikationsbarrieren zu überwinden, und zum anderen die Kommunikation analysiert und unterstützende Schlussfolgerungen aus der Analyse zieht. Es ist bspw. vorstellbar, eingehende XML-Datenströme auf Stichwörter zu untersuchen und diese mit einer Datenbank für Synonyme abzugleichen. Gefundene Synonyme (aber auch Homonyme mit passender Übersetzungsvorschrift[605]) können dann direkt ins ursprüngliche XML-Dokument durch geeignete Tags integriert werden.

Suchsystem

Das Suchsystem übernimmt alle an die Datenbanken gerichteten Anfragen. Dies können eigene, interne Anfragen des Systems sowie externe Anfragen von Kunden sein. Bei externen Anfragen wird es sich vorwiegend um von Kunden initiierte, die Komponentensuche betreffende Anfragen handeln. Interne Anfragen sind bspw. die Suche der Darstellungskomponente nach Darstellungsformaten für ein spezielles Endgerät. In der Abbildung 56 sind die internen Abfragen schematisch so dargestellt, als würden sie direkt auf die Datenbank zugreifen. Dies ist nicht ganz korrekt, aber für die Dar-

[604] Vgl. Lehner 2002.
[605] Vgl. Ortner 1993, S. 22.

stellung zweckmäßig, da man so erkennen kann, welche Komponente von welcher Datenbank Informationen erhält.

Der Grund für die zentrale Koordination aller Anfragen an das Informationssystem liegt darin, dass bei Änderung der Datenbankstruktur nur die Anfragen in der Komponente Suchsystem verändert werden müssen – analog zur Implementierung im CompoNex-Marktplatz[606]. Als Komponentensprache bietet sich z. B. XML Query an, wenn man von XML als allgemeiner Kommunikationssprache zwischen den Komponenten ausgeht.

Die Datenbasis für das Suchsystem liegt in der Komponenten-Datenbank (DBKat) vor. Aus dieser Datenbank wird ein stets aktuelles Komponenten-Repositorium (siehe Abschnitt 4.4.2) konstruiert, auf das die Anwender in Form eines Komponentenkatalogs Zugriff haben. Dieser Katalog stellt sich individuell auf die Anwender ein, denn die Vorfilterung wird bei jedem Anwender andere Komponenten ein- bzw. ausschließen. Der dynamisch erzeugte, individuelle Katalog wird wie die Einzelkomponente nach den Aspekten Vermarktung, Qualität, Aufgabe und Terminologie strukturiert. Die Aspekte Abstimmung, Verhalten und Schnittstelle werden wegen ihres rein technischen Aspekts nur im Vorfeld der Filterung benötigt. Jeder Aspekt bildet einen eigenen Hierarchiebaum aus. Da der Aufgabenaspekt eng mit dem Endgerätetyp korrespondiert, sollte auch dessen Klassifikation mit aufgenommen werden.

Dieser Katalog kann nun auf zwei Arten nach Komponenten durchsucht werden: zum einen über eine Suche mit Hilfe von Suchphrasen und zum anderen über ein Browsen (dt. Stöbern) im Katalog (siehe Abschnitt 4.4.2.2). Beide Strategien haben ihre Schwachstellen. Die Phrasensuche hat Probleme mit der Transformation der vom Anwender geäußerten Bedürfnisse in gezielte Abfragen der Datenbank – die Katalogsuche wiederum kann wegen der schwierigen Klassifikation von Komponenten in Domänen des Alltags auch nicht in jedem Fall ausreichend exakte Ergebnisse liefern.

Das System muss deswegen neben einer Vorfilterung technisch nicht integrierbarer Komponenten paralleles Suchen und Browsen unterstützen. Bei der Suche müssen gängige Methoden wie die Schlagwortsuche, die Volltextsuche, die Suche in auswählbaren Aspekten der Spezifikation etc. unterstützt werden. Vernetzte Komponentenbeschreibungen (bspw. durch Links wie in HTML-Seiten) unterstützen den Anwender

[606] Vgl. Ortner; Overhage 2003b, S. 30.

beim Browsen. Ein Subskriptionsdienst kann mit einer einmal eingegebenen Suchphrase in regelmäßigen Abständen den Katalog durchsuchen und den Anwender benachrichtigen, wenn eine neue Komponente die Suchkriterien erfüllt[607].

An dieser Stelle sei angemerkt, dass das Suchsystem lediglich den technischen Zugang zu den Datenbanken auf eine höhere Abstraktionsebene stellt und den anderen Komponenten somit den Zugriff auf alle Datenspeicherungssysteme erleichtert. Es ist explizit nicht Aufgabe dieses Systems zu „erraten", welche Informationen denn tatsächlich von anderen Komponenten oder bei externen Anfragen vom Anwender benötigt werden. Im letzteren der beiden Fälle ist dies Aufgabe des Reasoning-Management-Systems. Die Suchkomponente hat nur dafür Sorge zu tragen, dass die Anfragen syntaktisch richtig in der Sprache der Datenbank(en) formuliert werden und die Antworten wieder zurückübersetzt dem anfragenden System zur Verfügung gestellt werden. Ob die Antworten inhaltlich dem anfragenden System „nützen" liegt nicht in der Hand des Suchsystems. Diese Feststellung ist von Bedeutung, da das Auffinden geeigneter (Software-)Komponenten wahrscheinlich der entscheidende Faktor für den Erfolg eines Marktplatzsystems ist[608] (siehe Abschnitt 4.4.2.2).

Vorgangssteuerungssystem

Das Vorgangssteuerungssystem (engl. Workflow-Management-System) steuert zentral alle Prozesse des Marktplatzsystems. Es sorgt dafür, dass alle Marktphasen in geeigneter Weise abgebildet und unterstützt werden. In der Informationsphase muss das Darstellungssystem Informationen zum Komponentenkauf in geeigneter Art und Weise bereitstellen. Die vertraglichen Regelungen werden in der Vereinbarungsphase getroffen. Hierbei sorgt das Vorgangssteuerungssystem für die Einhaltung der notwendigen Prozessschritte und stellt die relevanten Informationen über das Darstellungssystem zur Verfügung. Während der Abwicklungsphase müssen die Vereinbarungen bei der Abwicklung eines geschlossenen Vertrages eingehalten werden. Nach der Feststellung, dass ein Kaufvertrag abgeschlossen wurde, muss die Zahlung des Kaufpreises erfolgen. Erst danach wird im Regelfall die Komponente zum Anwender transferiert. Andere Abwicklungsmodalitäten sind auch möglich und können durch das Vorgangssteuerungssystem problemlos umgesetzt werden. Auch ein Marktplatzmodell, in dem das Marktplatzsystem nur Intermediär ist, also selbst keine Kontrolle über die Komponen-

[607] Vgl. Ortner; Overhage 2003b, S. 30.
[608] Vgl. Griffel 1998, S. 100.

tenauslieferung hat, kann durch einfache Änderungen im Prozessablauf modelliert werden.

Geht man von der Abwicklung über den Marktplatz aus, muss nach Feststellung des Zahlungseingangs durch das Marktplatzsystem der Transfer eingeleitet werden. Die Zahlungsinformation kann – je nach Geschäftsmodell und Art des Zahlungsverkehrs – von Kreditinstituten, dem Anbieter der Komponente oder aus internen Quellen kommen. Beim Transfer sind die Spezifika der mobilen Endgeräte, an die „ausgeliefert" wird, zu beachten. Dies geschieht durch die Komponenten Konfigurationssystem und Dateitransfersystem mit den jeweils angekoppelten Datenbanken.

Konfigurationssystem

Komponenten sind nicht sofort in jedes System integrierbar. Deswegen muss eine Komponente „fertig" in Form von mehreren Varianten oder „unfertig" als modifizierbare Komponente vorliegen (siehe Abschnitt 4.2.1). In der Anwendungsentwicklung bezeichnet man das anschließende Zusammensetzen der einzelnen Komponenten nach Stückliste oder Syntax zum Gesamterzeugnis (d. h. zur Anwendung) als Konfigurieren[609].

Aufgrund des Zusammenwirkens von Betriebssystem, bestehenden Anwendungen und Benutzereinstellungen ist das Konfigurieren keine leichte Aufgabe und kann sogar zu einem kombinatorischen Problem werden. Das vorliegende Konfigurationssystem versucht aufgrund der Zielgruppe der Konsumenten (übermittelte (Software-)Komponenten sollen als Handelsware aufgefasst werden) trotzdem, diese Aufgabe automatisiert abzuwickeln. Dazu muss es auf Informationen in verschiedenen Datenbanken zurückgreifen. Zum einen kommen Informationen über die Hard- und Softwareausstattung des Endgerätes von der Endgeräte-Datenbank (DBMeg), zum anderen die persönlichen Einstellungen aus der Benutzerdatenbank (DBUser). Ein weiterer Nutzen kann aus der Transaktionen-Datenbank (DBTrans) gezogen werden, denn diese speichert die bereits vom Marktplatzsystem bezogenen Komponenten. Die neue Komponente kann dann im Hinblick auf die bereits verfügbaren Komponenten weiter optimiert werden. Dem Abschluss der Konfiguration folgt die Weiterleitung an das Dateitransfersystem.

[609] Vgl. Ortner 2005a, S. 115.

Dateitransfersystem
Nachdem die Komponente (ablauffähig) für das Endgerät des Kunden konfiguriert wurde, muss es zu diesem – evtl. verschlüsselt und signiert durch das Signatur- und Verschlüsselungssystem – übertragen werden. Dabei ist das Dateitransfersystem für die vollständige und fehlerfreie Übertragung der Komponenten zuständig. Da die Komponente in der Regel auf eine mobile Einheit heruntergeladen wird, die auch während des Dateitransfers nicht stationär ist, muss mit Übertragungsstörungen bis hin zum Abbruch der Verbindung gerechnet werden. Das Dateitransfersystem sollte deshalb als Mindestanforderungen die vollständige und fehlerfreie Übertragung mit der Wiederaufnahme von abgebrochenen Übertragungen (Resuming) unterstützen (siehe Abschnitt 4.4.6).

Signatur- und Verschlüsselungssystem
Um die Komponenten bei der Übertragung vom Marktplatz zum Kunden gegen Manipulationsversuche zu schützen, kann die Kommunikation verschlüsselt erfolgen oder die zu übermittelnde Komponente verschlüsselt und signiert werden. Eine Signatur bietet zudem für alle Beteiligten Rechtssicherheit beim Abschluss von Kaufverträgen in der Vereinbarungsphase. Das Beachten von Sicherheitsmerkmalen stärkt das Vertrauen der Handelspartner in den Marktplatz.

Das Signatur- und Verschlüsselungssystem ist eine eigenständige Komponente, die parallel von der Kommunikations- und Dateitransferkomponente eingesetzt werden kann. Die Betonung der Eigenständigkeit ist wichtig, denn sollte sich ein kryptographisches Verfahren als unsicher herausstellen, muss es schnellstmöglich durch ein sicheres ersetzt werden.

Metainformationssystem/Repositorium
Als zentrale Komponente des Marktplatzsystems steht das Metainformationssystem als integrierende Komponente für das Daten- und Metadatenmanagement stellvertretend für eine Vielzahl an Datenbanken, dem Repositorium und einer Management-Komponente.

Die einzelnen Datenbanken werden in ihrer tatsächlichen Implementierung unterschiedlich sein, da sie verschiedenen Ansprüchen genügen müssen. Die Lexikon-Datenbank (DBLex) etwa, die Synonyme verwaltet und damit für das Reasoning-Management-System von besonderer Bedeutung ist, muss die Daten aufgrund des zu

erwartenden hohen Kommunikationsanteils äußerst schnell zur Verfügung stellen können. Die Datenbank, die zur Abrechnung von erbrachten Leistungen eingesetzt wird (engl. Billing), muss dagegen in erster Linie für eine sichere und fehlerfreie Datenhaltung sorgen.

Auf die Datenbanken setzt das Metainformationssystem bzw. das Repositorium auf. Das Metainformationssystem kann, wie im E-NOgS³, durchaus als generisches System implementiert sein. Es stellt Informationen über Informationen zur Verfügung. Dies kann bspw. das Erstellungsdatum bestimmter Datensätze bzw. ganzer Tabellen sein. Es dient als Orientierungshilfe falls nicht klar ist, in welcher Datenbank Informationen überhaupt gesucht werden müssen. Das Repositorium enthält Beschreibungen von Informationsstrukturen, die besonders für vorgefertigte und häufig wiederkehrende Abfragen von Bedeutung sind. Typische Informationsstrukturen auf Marktplätzen sind Informationen über den Aufbau des Komponentenkatalogs, den Anwender später durchsuchen können. In diesem Fall werden die verschiedenen Komponentenaspekte dynamisch aus der Datenbank erzeugt und nach Vorgabe der Informationen im Repositorium dargestellt (siehe Abschnitt 4.4.2).

Es folgt eine Beschreibung aller Datenbanken, die in der Referenzarchitektur Verwendung finden.

- *DBKat*
  Eine Komponente kann unter verschiedenen Sichtweisen bzw. Perspektiven beschrieben werden. Diese sind aber bei der Speicherung der Daten in der Datenbank nicht von Interesse – hier geht es vielmehr um effiziente Speicherverfahren. Auf Grundlage der Komponentenspezifikation (siehe Abschnitte 3.5.2.6 und 4.4.2.1), die als Metainformation Teil des Repositoriums ist, wird der Katalog dann dynamisch mit Hilfe der in DBKat gespeicherten Daten rekonstruiert.

- *DBTrans*
  Die Transaktionen-Datenbank speichert Daten, die auf Transaktionen zwischen Marktplatzsystem und Kunden oder Marktplatzsystem und Anbieter basieren. Transaktionen sind alle Vorgänge, die von ökonomischer und technischer Relevanz für den Marktplatzbetreiber oder die beteiligten Partner sind. Dies ist zum einen der konkrete Transfer von Komponenten zum Anwender, aber – je nach Geschäftsmodell – z. B. auch schon die Bereitstellung von Informationen.

- *DBMeg*
  Da mobile Endgeräte über eine Vielzahl unterschiedlicher Konfigurationen verfügen, muss das Marktplatzsystem bei der automatischen Modifikation und Konfiguration von (Software-)Komponenten Kenntnis über diese haben. In der Datenbank DBMeg sind Informationen über die Bildschirmauflösung und Bildschirmfarbtiefe, aber auch unterstützte Kommunikationsprotokolle (z. B. WAP und HTTP) für jedes mobile Endgerät verfügbar. Die Daten sind vom Anwender nicht beeinflussbar und können evtl. automatisiert von den jeweiligen Herstellern mobiler Endgeräte bezogen werden. Auch Transformationsvorschriften für neue Protokolle, die aus bekannten Protokollen abgeleitet und vom Darstellungssystem verwendet werden, können hier abgelegt und vom Repositorium verwaltet werden. Benutzerspezifische und -definierbare Einstellungen werden in der Benutzerdatenbank gespeichert.

- *DBUser*
  Die Benutzerdatenbank DBUser speichert Parameter, die an den Anwender anpassbar oder von diesem vorgegeben werden. Das Spektrum der gespeicherten Informationen reicht von Adressinformationen über Daten zur Personalisierung der Marktplatz-Webseiten bis hin zu individuellen Konfigurationseinstellungen der verwendeten mobilen Endgeräte (siehe Abschnitt 4.4.3 und 4.4.4).

- *DBLex*
  Das Marktplatzsystem verfügt über ein Nachschlagewerk zur Unterstützung des Reasoning-Management-Systems. Hierin werden die semantischen Heterogenitäten erfasst. Es umfasst ein Lexikon von absolut notwendigen Fachbegriffen und verschiedene Wörterbücher zur Klärung von Sprachdefekten. Diese Wörterbücher sind für Anwender wichtig, da sie die Fachsprache von Software-Entwicklern nicht beherrschen (siehe Abschnitt 4.4.2.2) und Begriffe unterschiedlich einsetzen (Heterogenität).

Neben den aufgeführten Datenbanken ist eine Administrationskomponente (Datenbank-Management) wichtig, die dem Marktplatzbetreiber die Verwaltung der Datenbanken erlaubt. Diese Komponente ist meist durch eine einfache Konnektorkomponente, die eine Verbindung zum jeweiligen Datenbank-Managementsystem herstellt, implementiert.

5 mobiCOMP ein mobiler Marktplatz für den Handel von (Software-)Komponenten

Als Prototyp für die in Abschnitt 4.5 vorgestellte Referenzarchitektur mobiler Marktplatzsysteme wurde mobiCOMP entwickelt. mobiCOMP ist ein mobiler Marktplatz zum Handel von Software-Komponenten im mobilen Umfeld. Seine grundlegenden Funktionen sind das Anbieten, Suchen und Kaufen von Komponenten. Der Anbieter stellt seine Komponente und deren Beschreibung, die gemäß des GI-Spezifikationsrahmens[610] in einem Katalog abgelegt wird, in das Marktplatzsystem. Der Kunde kann Komponenten entweder über deren Namen oder hierarchisch über Kategorien suchen. Eine Detailansicht der Komponente liefert die durch den Anbieter bereitgestellte Beschreibung. Ein kaufwilliger Kunde kann dann die Komponente kaufen und herunterladen. Durch die Unterstützung von mobiler Kommunikationstechnologie ist dieser mobile Marktplatz von jedem Ort aus mit nahezu jedem möglichen mobilen Endgerät erreichbar. Durch eine geräteabhängige Anpassung der Darstellung werden die Komponenten gebrauchstauglich angeboten.

Das Projekt mobiCOMP entstand 2003 als Fortentwicklung der vergangenen zwei Wirtschaftsinformatikpraktika am Fachgebiet Wirtschaftsinformatik I der TU Darmstadt. Mit CompoNex[611] wurde 2001 ein elektronischer Marktplatz für Software-Komponenten entwickelt. Im darauf folgenden Jahr wurde mit mobileISM[612] eine Servicemanagementplattform für mobile Endgeräte auf PocketPC-Basis entwickelt. mobiCOMP bringt die mobile Welt und den Komponenten-Handel in einem integrierten Marktplatzsystem zusammen.

In dem hier dokumentierten Entwicklungsstand dient mobiCOMP zunächst als Marktplatz, über den allein lauffähige, unabhängige (Software-)Komponenten (Midlets, wie bspw. Spiele oder sogenannte Office-Anwendungen) angeboten werden. Als präsentationstaugliche Midlets wurden ein Spiel (mobiBreak – ein einfaches Tennisspiel) und ein Deutsch-Englisch-Wörterbuch (mobiDictionary) entwickelt.

---

[610] Vgl. Ackermann et al. 2002.
[611] Vgl. Ortner; Overhage 2003b.
[612] Vgl. Repp 2003.

Abbildung 57: mobiCOMP - Hardware[613]

Als mobile Marktplatzzugangsgeräte werden PDAs, die einen Internetbrowser (zur Anzeige von HTML-Seiten) bereitstellen und über eine Verbindung mit dem Internet verfügen sowie alle Mobiltelefone mit einem WAP-Browser (zur Anzeige von WML-Seiten) unterstützt. Als Kommunikationskanal dienen Internet (LAN/WLAN), WAP über GSM, GPRS und UMTS sowie serielle Bluetooth-Verbindungen zur Nutzung einer Internetverbindung eines anderen Endgerätes (siehe Abbildung 57). Die Basisfunktionalitäten des hier vorgestellten Prototyps konnten auf der CeBIT 2004 präsentiert werden.

Bei den auf dem Prototyp angebotenen Komponenten handelt es sich um alleine lauffähige J2ME-Anwendungen. Somit können diese Komponenten auf allen Endgeräten, auf denen eine Java VM zur Verfügung steht, ausgeführt werden.

5.1 mobiCOMP-Architektur

Als Architekturgrundlage dient die Referenzarchitektur für elektronische Marktplätze für den (Software-)Komponenten-Handel im mobilen Umfeld, die in Abschnitt 4.5 vorgestellt wurde. Um mobiCOMP als Prototypen eines solchen Marktplatzsystems mit vertretbarem Aufwand realisieren zu können, wurden nicht alle Komponenten der Referenzarchitektur implementiert. Somit wurde auf die Realisierung des Signatur- und Verschlüsselungssystems, des Schlussfolgerungs-, Verständigungs- und Erwägungssystems und des Konfigurationssystems verzichtet. Eine weitere Vereinfachung der Architektur ergab sich durch Zusammenfassen des Kommunikations- und Dateitransfersystems. Das Suchsystem aus der Referenzarchitektur wurde in der mobiCOMP-Architektur fest in das Vorgangssteuerungssystem eingebettet, um eine einfache Katalog- und Stichwortsuche zu ermöglichen. Die Kapselung der Datenbankabfragen erfolgt mit der Komponente Datenbank-Wrapper.

---

[613] Lonthoff 2004, S. 456.

Die einzelnen Komponenten des mobiCOMP Marktplatzsystems wurden jeweils als eigenständige Web Services auf Microsoft .NET-Basis implementiert. Diese Kapselung bietet ein hohes Maß an Erweiterbarkeit und Wiederverwendbarkeit. Dadurch wird eine nahtlose Integration in die Anwendungsumgebung z. B. eines Unternehmens ermöglicht. Abbildung 58 stellt die Architektur von mobiCOMP dar.

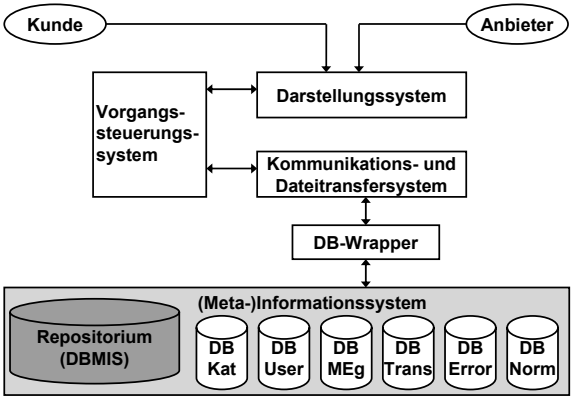

Abbildung 58: mobiCOMP - Software[614]

Die einzelnen, in Abbildung 58 aufgeführten Komponenten werden im Folgenden erläutert.

5.1.1 Das Darstellungssystem (DSS)

Das Darstellungssystem ist die generische grafische Benutzungsschnittstelle des Marktplatzsystems. Dieses System sorgt dafür, dass die Darstellung der zu übermittelnden Informationen auf das jeweilige Endgerät, das mit dem Marktplatz verbunden ist, angepasst wird. So wird ein Anwender, der über ein Mobiltelefon Zugang zum Marktplatz herstellt, mit Informationen im WML-Format versorgt, die dann der WAP-Browser des Endgerätes optimal darstellen kann. Für einen PDA, der einen viel kleineren Bildschirm besitzt als ein PC, wird ebenfalls eine reduzierte Darstellung im HTML-Format generiert. In Abbildung 59 wird auf der linken Seite eine Darstellung in einem Internet-Browser bei einer Auflösung von 1024 x 768 Pixel und auf der rechten Seite eine für PDA geeignete Darstellung mit 320 x 240 Pixel gezeigt. Um eine solche Anpassung der Darstellung zu ermöglichen, ist es notwendig, Darstellung und Inhalt zu trennen. Hierbei ist XML ein geeignetes Mittel. Über eine Transformation

[614] Lonthoff 2004, S. 457.

mittels XML Stylesheet Language Transformation (XSLT) können dann die Inhalte gefiltert und mittels Stylesheets auflösungsabhängig transformiert werden. Die anzuzeigenden Bildschirmelemente werden dann in Formularen angeordnet und gespeichert.

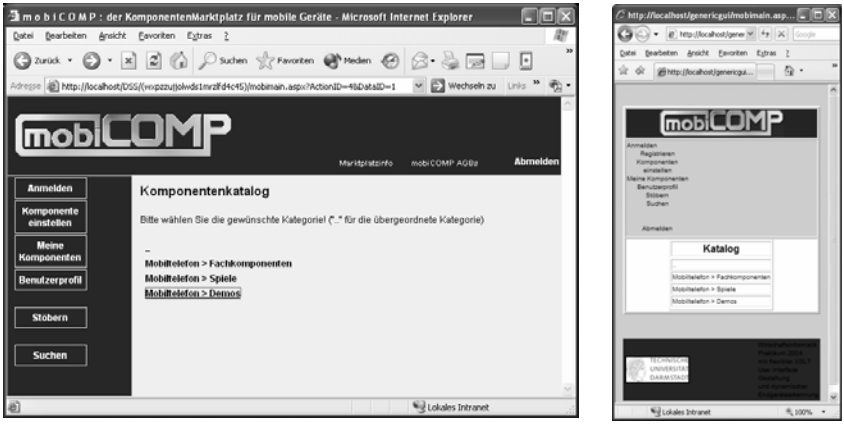

Abbildung 59: Darstellungsformen von mobiCOMP in einem Internet Browser[615]

Darüber hinaus sorgt das DSS für eine Überprüfung der getätigten Eingaben. Überprüft wird der Feldtyp, also der Wertebereich beim Ausfüllen der Felder und das Ausfüllen von Pflichtfeldern. Sollte mindestens eine der Eingaben ungültig sein, wird das Formular mit einem entsprechenden Hinweis erneut angezeigt. Sobald alle Eingaben korrekt vorliegen, werden diese an das Vorgangssteuerungssystem (VSS) weitergeleitet.

Alle Vorgaben zur Überprüfung der Eingaben werden den Metadaten der Felder entnommen. Die Felder werden in Formularen angeordnet. Die Formularmetadaten enthalten alle auf einem Formular anzuzeigenden Felder mit deren Metadaten. Diese Metadaten sind komplett in der Datenbank hinterlegt und werden jedem an das DSS gelieferte Objekt angefügt.

Aktionen des Anwenders werden an das VSS gesendet. Dieses verarbeitet die Aktion und sendet eine Formularidentifikationsnummer zurück. Daraufhin wird vom DSS das anzuzeigende Formular gerätespezifisch generiert.

---

[615] Vgl. Lonthoff 2004, S. 458, ergänzt um PDA-Darstellung.

5.1.2 Kommunikations- und Dateitransfersystem (KDS)

Das KDS ist das zentrale Kommunikationssystem. Es dient als einheitliche Kommunikationsschnittstelle zur Weiterleitung der Anfragen des Vorgangssteuerungssystems. Nach innen wird der geeignete Kommunikationszugang zum Marktplatzsystem gewährleistet. Nach außen sorgt das Dateitransfersystem für die Sicherstellung einer einwandfreien und vollständigen Übertragung der Komponente. Dafür muss das KDS feststellen können, ob ein Transfer vollständig abgewickelt wurde. Im Fehlerfall muss das VSS informiert werden.

5.1.3 Vorgangssteuerungssystem (VSS)

Das VSS übernimmt die Koordination und Steuerung aller möglichen Aktionen des Marktplatzsystems in allen Handelsphasen.

In der Informationsphase müssen die notwendigen Informationen zum Finden von Komponenten zur Verfügung gestellt werden. In der Verhandlungsphase besteht die große Herausforderung in einer Rechnerunterstützung der Vertragsverhandlung. In der Abwicklungsphase muss das VSS dafür sorgen, dass alle relevanten Arbeitsschritte bei der Abwicklung eines geschlossenen Vertrages eingehalten werden.

Nach der Feststellung, dass ein Kaufvertrag abgeschlossen wurde, muss die Zahlung des Kaufpreises erfolgen. Die Zahlungsinformation kann – je nach Geschäftsmodell und Art des Zahlungsverkehrs – von Kreditinstituten, dem Verkäufer der Komponente oder aus internen Quellen kommen. Nach erfolgter Zahlung wird dann im Regelfall die Komponente zum Kunden transferiert. Dieser Transfer muss dann auch sichergestellt sein. Ein Verbindungsabbruch darf nicht zu einem Verlust der erworbenen Komponente beim Kunden führen, sondern es müssen evtl. Wiederholungsversuche des Komponententransfers gestattet sein. Dabei ist der Komponententransfer Aufgabe des KDS. Das VSS übernimmt also die Ablauflogik des Systems.

5.1.4 Datenbank-Wrapper (DBW)

Der Datenbank-Wrapper kapselt den Zugriff auf die Datenbanken. Der Grund für die zentrale Koordination aller Abfragen an das (Meta-)Informationssystem liegt darin, dass in diesem Fall eine Änderung der Datenbankstruktur nicht zu Modifikationen des Programmcodes führt. Der DBW stellt drei Basisfunktionen zum Einfügen, Auswählen und Löschen (Insert, Select und Delete) zur Verfügung. Dabei werden die Tabellenstrukturen und die Objekte aus XML-Dateien ausgelesen, um die Abfragen entsprechend generisch aufzubauen.

Angemerkt sei an dieser Stelle, dass der DBW lediglich den technischen Zugang zu den Datenbanken auf eine logische, abstraktere Stufe stellt und den anderen Komponenten des Marktplatzsystems somit den Zugriff auf alle Datenspeicherungssysteme erleichtert. Er hat dafür Sorge zu tragen, dass die Anfragen in der Sprache der Datenbank(en) syntaktisch korrekt formuliert sind und die Antworten der Datenbank-Systeme in die Sprache des anfragenden Systems zur Verfügung gestellt werden.

5.1.5 (Meta-)Informationssystem

Die Komponente, die in der funktionalen Übersicht mit (Meta-)Informationssystem bezeichnet ist, besteht aus einem Repositorium (DBMIS), in dem die Metadaten abgelegt werden und den Datenbanken DBKat, DBUser, DBMeg, DBTrans, DBError und DBNorm.

DBKat – Komponentenkatalog-Datenbank

DBKat umfasst die zur Umsetzung des Komponentenspezifikationsrahmens der GI[616] (siehe Abschnitt 3.5.2.5) notwendigen Relationen. Es wurde eine möglichst weitgehend mit der Theorie übereinstimmende Realisierung angestrebt, sodass das vorliegende Marktplatzsystem zugleich als Machbarkeitsstudie des vorgeschlagenen Spezifikationsrahmens gelten kann.

DBUser – Benutzer-Datenbank

DBUser speichert sämtliche dem System bekannten natürlichen und juristischen Entitäten strukturiert ab. Hierbei wird zwischen natürlichen und juristischen Personen unterschieden. Natürliche Personen (Datenstruktur „Person“) können sowohl als Systemanwender (verbunden mit einem Login-Account) als auch als Ansprechpartner für Unternehmen (Datenstruktur „ContactPerson“ als Resultat der Konnexion mit „Company“) gelten. Juristischen Personen dienen zur Identifikation des Herstellers der Komponenten.

DBMeg – Endgeräte-Datenbank

DBMeg dient der strukturierten, hierarchischen Gliederung der vom Marktplatzsystem unterstützten mobilen Endgeräte. Neben der Einordnung des jeweiligen Gerätetyps unter eine der hierarchisch gegliederten Geräteklassen (Mobiltelefon, Smartphone, PDA) werden Informationen, wie Bildschirmauflösung und Bildschirmfarbtiefe, abgelegt, die zur Darstellung der Marktplatz-Bedienumgebung auf dem Endgerät sowie der

[616] Vgl. Ackermann et al. 2002, S. 4.

Unterstützungsfunktionalität bezüglich des Komponententransfers von anderen Marktplatzkomponenten verwendet werden.

DBTrans – Transaktionen-Datenbank
DBTrans dient als operative Logbuch-Datenbank. Alle systemrelevanten Ereignisse werden hier chronologisch – zu Debugging-Zwecken, als Datenbasis für Statistiken und für Sicherheitsabfragen (bspw. versuchtes Mehrfach-Einloggen) etc. – protokolliert.

Systemrelevante Ereignisse lassen sich untergliedern in anwender-, komponenten- und systembezogene Ereignisse. Anwenderbezogene Ereignisse sind Login und Logout von Anwendern, Registrierung und Abmeldung vom Marktplatz sowie Änderungen von Anwenderdaten. Komponentenbezogene Ereignisse sind das Auswählen der Detailansicht einer Komponente, Kauf und Herunterladen von Komponenten, Einfügen, Ändern und Löschen von Komponenten. Eine strukturierte Gliederung der im Gesamtsystem möglichen Fehlerfälle ist den systembezogenen Ereignissen zuzuordnen (siehe hierzu auch DBError).

DBError – Fehler-Datenbank
Ziel von DBError ist die zentralisierte Administration sämtlicher potenziell möglicher Fehlerereignisse. Hierbei werden die Fehlerereignisse baumstrukturartig in festgelegte Fehlerklassen gegliedert.

DBNorm – Normsprachenschema
Die Normsprachenebene stellt eine terminologiebasierte Funktionalität für zwei Sprachebenen bereit. Auf der objektsprachlichen Ebene werden Termini und Satzbaupläne zur Beschreibung von im Marktplatz gehandelten Komponenten abgelegt, auf der Metaebene hingegen werden metasprachliche Konstrukte (Attribute, Relationen) Termini zugeordnet, sodass u. a. transparente Synonymitätsauflösung ermöglicht wird.

Zum Erreichen o. g. Ziele erfolgt eine explizite, schrittweise und zirkelfreie Definition von Fachbegriffen (Termini), welche nicht mit den weiter unten aufgeführten Schlüsselbegriffen (Keywords) zu verwechseln sind. Während erstere in der Aufgabenebene verwendet werden und Datenstrukturen von Komponenten normalsprachlich (gebrauchssprachlich) bezeichnen, dienen Schlüsselbegriffe der semantischen Suche nach Komponenten. Definiert werden Termini nach den gängigen Definitionsregeln – neben

obligatorischer Kurzdefinition sind Langdefinitionen, Prädikatorenregeln und Verwendungsbeispiele möglich. Des Weiteren können Termini zueinander in Beziehung (z. B. „ist Gattung von“ oder „ist Schlüssel für“) stehen.

Objektsprachliche Termini können entweder global – dies ist insbesondere zur Standardisierung und der daraus resultierenden Schaffung von Interoperabilität wichtig (siehe Abschnitt 3.1.1) – oder lokal für einzelne Komponenten gelten. Metasprachliche Termini sind in diesem Sinne stets als global anzusehen.

DBMIS – Metainformationssystem
DBMIS stellt Metainformationen, d. h. Informationen über die im System existierenden Schemata (Konzepte wie Relationen, Attribute, Integritätsbedingungen etc.) strukturiert dar. Hierbei erfolgt eine transparente Auflösung von Synonymen auf Attributebene durch Bezugnahme auf normsprachliche Daten (DBNorm).

Der vollständige Systementwurf von mobiCOMP findet sich im Anhang.

5.2 Anwendungsszenarien von mobiCOMP

Für einen mobilen Marktplatz als zentralen Ort der Komponentenbeschaffung lassen sich verschiedene Anwendungsszenarien skizzieren. Vorstellbare Anwendungsszenarien reichen vom Marktplatz für Midlets (wie mit mobiCOMP bereits realisiert) bis zum automatisierten Handel und Konfigurieren von Fachkomponenten für den Aufbau komplexer Anwendungssysteme. Darüber hinaus wäre die Eignung anderer Nutzungsarten, wie in Abschnitt 4.3.1 beschrieben zu untersuchen.

Szenario „Spiele und kleine Anwendungen“:
In diesem Szenario werden auf mobiCOMP sogenannte Midlets gehandelt. Das sind J2ME-Anwendungen, wie Spiele, Nachschlagewerke oder Stundenplaner. Der Markt dafür ist im Konsumentenbereich vorhanden, d. h. es gibt schon etliche Anbieter (z. B. Jamba[617]) und die Nachfrage ist groß. Der Vorteil von mobiCOMP besteht darin, dass das Produkt nicht wie bspw. bei Jamba über teure 0900er Nummern oder umständliche Codes zu erhalten ist, sondern entweder direkt per WAP-Portal oder bspw. über einen Bluetooth-Hotspot in einem Telekommunikationsgeschäft oder einer Tankstelle.

[617] www.jamba.de

Szenario „mobile Business“:
Auf dem Marktplatz werden in diesem Szenario einfache Fachkomponenten für mobile Endgeräte gehandelt. Dabei werden die Komponenten-Suche und der Komponenten-Handel explizit vom Anwender (Kunde) veranlasst. Der Marktplatz ist reiner Intermediär i. S. des Handels. Auf einem solchen Marktplatz können z. B. Komponenten für den mobilen Zugang zu Enterprise Resource Planning (ERP) Systemen oder Komponenten für die Abrechnung von mobilen Dienstleistungen gehandelt werden. In diesem Szenario ist der Anwender selbst verantwortlich für eine geeignete Integration der transferierten Komponente zu sorgen.

Szenario „mobiles externes Anwendungsmanagement“:
In diesem Szenario werden benötigte (Software-)Komponenten automatisiert vom Marktplatzsystem abgerufen, gerätespezifisch konfiguriert und auf dem mobilen Endgerät installiert. In diesem Szenario ist das Marktplatzsystem das externe Anwendungsmanagement im Sinne der Bereitstellung von Komponenten für ein mobiles Endgerät. Dabei kann eine Versionskontrolle den gesamten Lebenszyklus der angebotenen Komponenten überwachen und gegebenenfalls aktualisierte Komponenten bereitstellen. Hierbei wird ein internes Anwendungsmanagement auf dem mobilen Endgerät vorausgesetzt.

5.3 Weiterentwicklung von mobiCOMP

Eine rechnerunterstützte benutzungsfreundliche Bereitstellung von Anwendungen auf Komponentenbasis ist ein hochgestecktes Ziel. Dabei soll benutzungsfreundlich so verstanden werden, dass ein Anwender, der kein spezielles Vorwissen im IT-Umfeld hat, den Marktplatz nutzen kann und durch das Angebot auf dem Marktplatz unterstützt wird.

Das in mobiCOMP realisierte generische Darstellungssystem sowie das Abbilden der Komponentenbeschreibung nach dem Spezifikationsrahmen der GI führen durch eine hohe Anreicherung des Informationsflusses mit Metadaten zu einem erhöhten Kommunikationsaufwand. In stationären Netzwerken stellt dies kein wahrnehmbares Problem dar. Aber durch die Nutzung mobiler Endgeräte, die auf den Marktplatz unter Verwendung drahtloser Netzwerke zugreifen, wirkt sich das erhöhte Datenvolumen negativ auf die Nutzung des Marktplatzes aus. Der Anwender nimmt deutliche Verzögerungen in der Interaktion mit dem Marktplatzsystem wahr.

Eine Erleichterung für den Anwender ist das automatische Vorfiltern des Angebots. Dies kann z. B. dadurch geschehen, dass beim Zugriff auf den Marktplatz das Endgerät des Anwenders identifiziert wird und automatisch Komponenten für diese Gerätekategorie vorgeschlagen werden. Bei dieser Erweiterung sollte eine explizite Suche nach ausgeblendeten Angeboten trotzdem noch möglich sein, falls das Endgerät falsch identifiziert wurde oder der Anwender explizit etwas anderes sucht.

Als Erweiterung der implementierten Stichwortsuche sollte eine Unterstützung durch eine semantische Suche realisiert werden, die das gezielte Auffinden von Komponenten insbesondere im Bereich mobiler Endgeräte gestattet. Das Ausfüllen einer komplexen Suchanfrage ist mit Hilfe der Eingabeschnittstellen, die bei mobilen Endgeräten zur Verfügung stehen, nur umständlich durchführbar. Stichwörter besitzen eine gewisse „Nähe“ zueinander, sodass von eingegebenen Stichwörtern zu „Nachbarn“ gemäß den im System abgelegten Distanzmaßen übergegangen werden kann (fuzzy search). Dies wäre eine sinnvolle Erweiterung der Suche bei expliziten Anfragen durch den Anwender.

Ein Marktplatz kann hierbei verschiedene Rollen einnehmen. Als Intermediär ist er z. B. zwischen Anbieter und Kunden anzuordnen. Damit nimmt er die Funktion einer Anbieter-Nachfrager-Zusammenführung wahr, ohne dass Anbieter und Kunden direkt zusammentreffen. Im Sinne eines Integrators[618] tritt er dann auf, wenn durch das Marktplatzsystem unabhängige, physisch und logisch getrennte, meist heterogene Systeme miteinander verbunden werden. Wenn die Funktionen der gehandelten Komponenten als Dienste bereitgestellt werden, wird aus dem Marktplatzsystem ein sogenannter Trader[619], da die Dienste in einem verteilten System vermittelt werden können. Darüber hinaus kann ein Marktplatz durch Bereitstellung von zusätzlichen Diensten als Servicecenter einen Mehrwert bereitstellen.

Das Ziel einer generischen, regelbasierten Marktplatzsteuerung konnte noch nicht erreicht werden. Für eine flexible Ablaufsteuerung ist das Vorgangssteuerungssystem derart zu erweitern, dass es ausschließlich generischer Natur ist und die anzuwendenden Regeln (Prozessschemata) einer Datenbank entnimmt. Durch eine solche flexible Steuerung des Marktplatzsystems können auch unterschiedliche Marktformen und

[618] Vgl. Voigtmann; Zeller 2003, S. 222.
[619] Vgl. Eicker; Schwichtenberg 2003, S. 303.

Marktmechanismen unterstützt werden ohne das Gesamtsystem modifizieren zu müssen.

Um die Qualität der angebotenen Komponenten zu gewährleisten, sollte ein Prüfstand (ähnlich einem „TÜV“) für Komponenten entwickelt werden. In Verbindung mit der Vergabe von Zertifikaten für geprüfte Komponenten, kann somit das Vertrauen durch Gewährleistung von Qualitätsmerkmalen gesteigert werden.

Für die automatisierte Komposition von Komponenten wird eine spezifische Komponente (Composer) benötigt. Diese hat zur Aufgabe, die Komponenten anwendungs- und endgerätespezifisch (nach einem übergeordneten Bauplan) zusammenzustellen und so zu konfigurieren, dass das Kompositum automatisch auf dem Endgerät installiert wird. Ein Marktplatz, der diese Automatisierung von Konfiguration und Installation beherrscht, kann den kompletten Komponenten-Lebenszyklus i. S. eines Anwendungssystem-Management-Systems managen, in dem er auch noch Funktionen zum automatischen Deinstallieren, bzw. Aktualisieren von Komponenten anbietet.

Im Rahmen eines ganzheitlichen Sicherheitsmanagements sollte in mobiCOMP ein Sicherheitskonzept umgesetzt werden, das die grundlegenden Schutzziele Vertraulichkeit, Authentizität, Integrität, Nichtabstreitbarkeit und Verfügbarkeit gewährleistet. Hierzu müsste die beim Prototyp weggelassene Komponente „Verschlüsselungssystem“ auf Basis aktueller kryptografischer Methoden implementiert werden. Das Verschlüsselungssystem ermöglicht dann eine sichere Übertragung der Komponenten und eine geeignete Authentifizierung der Marktplatz-Teilnehmer.

Eine orthogonale Funktionserweiterung von mobiCOMP kann dahingehend erfolgen, dass die Lokalisierungsmöglichkeit von mobilen Endgeräten mit berücksichtigt wird und eine entsprechende Lokalisierungsnutzung (siehe Abschnitt 4.4.7) erfolgt.

6 Schlussbetrachtung

Mit dieser Arbeit liegt ein weiterer Schritt zur Entwicklung komponentenbasierter Anwendungssysteme vor. Komponentenbasierte Anwendungssysteme sind dadurch gekennzeichnet, dass sie schon bei der Entwicklung modular und arbeitsteilig modelliert und konstruiert werden. Ein Großteil des Wissens fließt hierbei in die Modellierung (Spezifikation) der Anwendungsarchitekturen und deren (Software-)Komponenten ein. Eine nachgelagerte Implementierung kann entweder in Billiglohnländern (Near- bis Offshoring) oder zunehmend automatisiert durch Code-Generatoren erfolgen.

Komponentenbasierte Anwendungssysteme werden aus vorgefertigten, ggf. aber auch individuell hergestellten, wieder verwendbaren (Software-)Komponenten aufgebaut. Diese können in einem Komponentenkatalog (oder Repositorium), der als ein elektronischer Marktplatz implementiert sein kann, abgelegt und dokumentiert werden. Die Komponenten können daraufhin aus dem Marktplatz zwecks Einsatzes in einem Anwendungssystem entnommen werden.

Die Betrachtung von komponentenbasierten Anwendungssystemen ließ zwei voneinander unabhängige Lebenszyklen erkennen: einen Lebenszyklus der Entwicklung von Anwendungssystemen (Vorgehensmodelle) und einen Lebenszyklus der Komponenten eines Anwendungssystems. Das Anwendungsmanagement – wie es in dieser Arbeit vorgestellt wurde – plant, steuert und kontrolliert Aufgaben dieser beiden Lebenszyklen.

Eine weitere Unterteilung des Anwendungsmanagements kann anhand des Aspekts der Ressourcenherkunft in ein internes und externes Anwendungsmanagement erfolgen. Alle entwicklungs- und konfigurationsrelevanten Aufgaben werden dem internen Anwendungsmanagement zugeschrieben. Das externe Anwendungsmanagement hat die Aufgabe, (Software-)Komponenten von außen z. B. von einem (Software-)Komponenten-Marktplatz oder als Dienst bspw. in Form von Web Services in den Anwendungssystem-Lebenszyklus zu integrieren.

(Software-)Komponenten-Marktplätze leisten einen wichtigen Beitrag zum Aufbau effizienter Anwendungssysteme aus (global) gehandelten Komponenten. Anbieter der Software-Branche können ihr Wissen dazu nutzen, bessere Komponenten für den globalen Komponentenmarkt zu entwickeln. Nachfrager nutzen entweder ihr internes

Management-System für Anwendungssysteme zur Konfigurierung ihrer Anwendungssysteme, oder sie beziehen das Anwendungssystem (oder auch Teile davon) von einem unabhängigen Dritten, z. B. dem Marktplatzbetreiber.

Nach einer breiten Untersuchung von Ansätzen des Anwendungsmanagements konnte unter Berücksichtigung der Eigenschaften des mobilen Umfelds ein Konzept für einen mobilen Marktplatz als Lösungsansatz für Aufgaben des externen Anwendungsmanagements entwickelt werden. Damit ein Handel der (Software-)Komponenten stattfinden kann, muss das immaterielle Gut „(Software-)Komponente“ sprachlich darstellbar sein. Andernfalls wäre eine (Software-)Komponente nicht vermarktbar.

Zur Überprüfung der konstruierten Referenzarchitektur für mobile elektronische Marktplätze eines (Software-)Komponenten-Handels wurde mit mobiCOMP ein Prototyp implementiert und somit eine Teillösung für das externe Anwendungsmanagement vorgestellt. Dabei wurden wesentliche Elemente der Referenzarchitektur für mobile (Software-)Komponenten-Marktplätze erfolgreich umgesetzt.

mobiCOMP wurde als Web Service auf der Basis von Mircosoft .NET implementiert. Zur Beschreibung der (Software-)Komponenten wurde der Spezifikationsrahmen für Fachkomponenten der Gesellschaft für Informatik eingesetzt. Auf Basis dieser Spezifikation wurde das Komponenten-Repositorium des Marktplatzsystems entwickelt.

Zwei Voraussetzungen müssen für mobiCOMP erfüllt sein, damit man von ihm als einer Implementierung des externen Anwendungsmanagements sprechen kann:

- mobiCOMP wird als zentraler Marktplatz-Server eingesetzt, der überall, d. h. ubiquitär vorhanden ist, indem auf ihn von beliebigen Orten zugegriffen werden kann.
- Es werden (Software-)Komponenten bereitgestellt, die vollständig und universell gemäß des implementierten Spezifikationsrahmens beschrieben sind.

Der ersten Forderung wurde damit Rechnung getragen, dass mobiCOMP zum einen Internet-basiert, also per Web-Browser zugänglich ist, und zum anderen, dass ein WAP-Portal für GSM- und UMTS-basierte Endgeräte bereitgestellt wird. Die zweite Forderung hat eher zukunftsweisenden Charakter, da globale universelle Standards, wie etwa in anderen Ingenieurbereichen hinsichtlich der Spezifikation ihrer Komponenten, in der Software-Branche noch nicht in ausreichender Form existieren.

Aus technologischer Sicht konnte in mobiCOMP ein Spezifikationsrahmen für Komponenten erfolgreich umgesetzt werden. Ebenso wurde die technologische Anbindung unterschiedlicher mobiler Endgeräte, die verschiedene drahtlose Netzwerke verwenden, realisiert. Durch das generische Darstellungssystem und die Komponentenbeschreibung wird wegen des hohen Metadatenanteils das zu übertragende Datenvolumen zwischen Anwender und Marktplatzsystem merklich erhöht. An dieser Stelle sollte der Einsatz geeigneter Kompressionsverfahren überprüft werden, um den Vorteil des flexiblen Darstellungssystems und der einheitlichen Komponentenspezifikation nicht zu beeinträchtigen. Im Bereich der semantischen Komponenten-Suche ist noch weitere Forschungsarbeit zu leisten.

Wenn (Software-)Komponenten interoperabel spezifiziert sind, d. h., wenn sie einer über Unternehmensgrenzen hinausgehenden einheitlichen Beschreibung folgen, und wenn ein internes Anwendungsmanagement auf jedem Endgerät verfügbar ist, dann leistet das externe Anwendungsmanagement einen wichtigen Beitrag zur Integration omnipräsenter Anwendungssysteme. Ein Erreichen der Universalität ermöglicht hierbei das Zwischensprachenkonzept, bei dem jede (Software-)Komponente ihre eigene „Sprache“ und eine festgelegte Zwischensprache beherrscht, welche von allen (Software-)Komponenten eines Anwendungssystems zum Zweck der Interaktion verwendet werden muss[620].

Von diesen Ergebnissen ausgehend ergeben sich verschiedene weitere Forschungsthemen im Umfeld komponentenbasierter, mobiler Anwendungssysteme, insbesondere in Verbindung mit einem integrierten internen Anwendungsmanagement. Ein daraus entstehendes Anwendungssystem-Management beinhaltet verschiedene Facetten, die es vor dem Hintergrund der Entwicklung moderner Informations- und Kommunikationstechnologie weiter zu erforschen gilt.

Grundsätzlich stellt sich die Frage, wann von einem mobilen Marktplatz gesprochen werden darf. Unter den Aspekten der Benutzer- und Dienstmobilität hat der Begriff seine Berechtigung. Somit ist ein mobiler Marktplatz ein elektronischer Marktplatz, der überall erreichbar ist, da auf ihn von beliebigen Orten mit mobilen Endgeräten zugegriffen werden kann.

[620] Vgl. Ortner 2005a, S. 213.

Die Zukunft der ingenieurmäßigen Softwareentwicklung ist komponentenbasiert. Doch es müssen noch Einschränkungen gemacht werden. Es gibt erst wenige (Software-)Komponenten, die einheitlich beschrieben auf Märkten gehandelt werden oder von ihrem Entwurf her überhaupt auf Märkten handelbar sind. Hier müssen auf Seiten der Anbieter und insbesondere im Bereich universeller Komponentenspezifikationen noch weitere Ergebnisse erarbeitet werden.

Höß stellt einige Arbeiten vor, die argumentativ darlegen, dass (Software-)Komponenten-Marktplätzen nicht funktionieren können[621]. Im Rahmen von SOA und der zunehmenden Industrialisierung der Anwendungssystementwicklung bleiben (Software-)Komponenten-Marktplätze weiterhin im Fokus der Anwendungsinformatik. Gerade im Hinblick auf omnipräsente Anwendungssysteme, die den Alltag des Menschen durchdringen werden, ist die wissenschaftliche und industrielle Beschäftigung mit dem Thema „Komponentenbasierung“ sinnvoll. Daher wurden in dieser Arbeit an dem Konzept der komponentenbasierten Anwendungssystementwicklung festgehalten und mögliche Nutzungsweisen entwickelt.

[621] Vgl. Höß 2005, Kapitel 2.4, S. 51ff.

Literaturverzeichnis

ANSI (2001): ATIS Telecom Glossary 2000, www.atis.org/tg2k/, Abgerufen am 28.06.06, 2001.

ANSI/X3/SPARC (1975): Study Group on Database Management – Systems-Interim-Report. In: Bulletin of ACM SIGMOD, Bd. 7, Nr. 2, 1975.

Ackermann, J. et al. (2002): Vereinheitlichte Spezifikation von Fachkomponenten. In: Memorandum des Arbeitskreises 5.10.3 der Gesellschaft für Informatik e.V. „Komponentenorientierte betriebliche Anwendungssysteme, Augsburg, 2002.

Alexander, C. et al. (1977): A Pattern Language: Towns – Buildings – Construction, 26. Aufl., New York, 1977.

Alexander, C. (1979): Center for Environmental Structure: The timeless Way of Building, New York, 1979.

Alexander, C. (1999): The Origins of Pattern Theory, the Future of the Theory, and the Generation of a Living World. In: IEEE Software, 1999, S. 71-82.

Andreou, A. S. et al. (2002): Mobile Commerce Applications and Services: A Design and Development Approach. In: Proceedings of the First International Conference on Mobile Business (M-Business 2002), Athen, 2002, S. 1-9.

Arildson, D.; Dedo, D. (2002): Pocket PC Systems Management - White Paper Microsoft, http://www.microsoft.com/technet/archive/itsolutions/mobile/evaluate/pctpcmgt.mspx?mfr=true, Abgerufen am 08.12.06, 2002.

Austin, D.; Barbir, A.; Garg, S. (2002): Web Services Architecture Requirements, http://www.w3.orgTR/wsa-reqs, Abgerufen am 28.06.06, 2002.

Austin, J. C. (1962): How to Do Things with Words, Cambridge, 1962.

Aßmann, U.; Neumann, R. (2003): Quo vadis Komponentensysteme? In: HMD - Praxis der Wirtschaftsinformatik, Nr.231, 2003, S. 19-27.

Babiak, U. (1999): Effektive Suche im Internet: Suchstrategien, Methoden, Quellen, 3. akt. u. erw. Aufl., Beijing [u. a.], 1999.

Balzert, H. (1989): Die Entwicklung von Software-Systemen: Prinzipien, Methoden, Sprachen, Werkzeuge, unveränd. Nachdr., Mannheim [u. a.], 1989.

Bass, L.; Clements, P.; Kazman, R. (2003): SEI Series in Software Engineering, Software Architecture in Practice, 2. Aufl., Reading [u. a.], 2003.

Beitz, A.; Bearman, M.; Vogel, A. (1995): Service Location in an Open Distributed Environment. In: Proceedings of 2nd Workshop on Services in Distributed and Networked Environments, Whistler, 1995, S. 28-34.

Bell, G.; Gray, J. N. (1997): The Revolution Yet to Happen. In: Denning, P. J.; Metcalfe, R. M. (Hrsg.): Beyond Calculation – The next fifty Years of Computing, New York, 1997, S. 5-32.

Berbner, R. et al. (2005): Management of Service-oriented Architecture (SoA)-based Application Systems. In: LNI-Proceedings of Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'05), Bd. 2, Nr. 1, 2005, S. 208-211.

Berkeley University of California (o. J.): SETI@home, www.setiathome.ssl.berkeley.edu/, Abgerufen am 18.04.06, o. J.

Bienert, P. (1998): Information und Kommunikation: Technik und Anwendung in Wirtschaft und Medien, Berlin [u. a.], 1998.

Biggerstaff, T. J.; Perlis, A. J. (1989): Software Reusability, Bd. 1, New York, 1989.

Bluetooth SIG (1999): Service Discovery Protocol (SDP), http://www.pday.com.cn/technology/bluetooth_documents/sdp.pdf, Abgerufen am 12.04.06, 1999.

Bluetooth SIG (2003): Specification of the Bluetooth System, www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2, Abgerufen am 05.06.06, 2003.

Bodendorf, F. (2006): Daten- und Wissensmanagement, 2. akt. u. erw. Aufl., Berlin [u. a.], 2006.

Bodoff, S. et al. (2002): The J2EE Tutorial, Indianapolis, 2002.

Boehm, B. (1988): A Spiral Model of Development and Enhancement. In: IEEE Computer 21, Nr. 5, 1988, S. 61-72.

Boutilier, C. et al. (2003): Cooperative Negotiation in Autonomic Systems using Incremental Utility Elicitation. In: Uncertainty in Artificial Intelligence, 2003, S. 89-97.

Brabänder, E.; Klückmann, J. (2006): Geschäftsprozessmanagement als Grundlage für SOA. In: OBJEKTspektrum, Nr. 5, 2006, S. 32-40.

Brandt, C.; Lonthoff, J. (2004): Vorschlag für eine Architektur zur Personalisierung und Individualisierung mobiler Dienste. In: Höpfner, H.; Saake, G. (Hrsg.): Workshop Grundlagen und Anwendungen mobiler Informationstechnologie, Preprint Nr. 4, Magdeburg, 2004, S. 33-40.

Broy, M. et al. (1998): What characterizes a (software) component? In: Software – Concepts and Tools, Bd. 19, Nr. 1, 1998, S. 49-56.

Buxmann, P. (1996): Standardisierung betrieblicher Informationssysteme, Dissertation Universität Frankfurt, Fakultät Betriebswirtschaftslehre, insbesondere Wirtschaftsinformatik, Wiesbaden, 1996.

Böhm, O. (2006): Aspektorientierte Programmierung mit AspectJ 5: Einsteigen in AspectJ und AOP, Heidelberg, 2006.

Capra, L.; Emmerich, W.; Mascolo, C. (2002): Middleware for Mobile Computing. In: LNCS-Proceedings Advanced Lectures on Networking: Networking 2002 Tutorials, Bd. 2497, Berlin [u. a.], 2002, S. 20-58.

Cerwenka, P. et al. (2000): Kompendium der Verkehrssystemplanung, Wien, 2000.

Chittaro, L. (2003): Human-Computer Interaction with Mobile Devices and Services, LNCS-Proceedings of 5th International Symposium on Mobile HCI 2003, Bd. 2795, Berlin [u. a.], 2003.

Clark, J. (1993): Human Resource Management and Technical Change, London [u. a.], 1993.

Coad, P.; Nicola, J. (1993): OOPObject-Oriented Programming, Englewood Cliffs, 1993.

ComponentSource (o. J.): The Open Market, http://www.componentsource.com/Services/TheOpenMarket.asp, Abgerufen am 11.04.06, o. J.

Councill, W. T.; Heinemann, G. T. (2001): Definition of a Software Component and its Elements. In: Heinemann, G. T.; Councill, W. T. (Hrsg.): Component-based Software Engineering, Upper Saddle River, 2001, S. 6-19.

Crnkovic, I.; Larrson, M. (2000): Component Configuration Management. In: Bosh, J. et al. (Hrsg.): Proceedings of the 5th Int. Workshop on Component-Oriented Programming (WCOP 2000). Research Report No 15/00, Blekinge Institute of Technology, Sophia Antipolis, 2000, S. 87-90.

Davenport, T. H.; Beck, J. C. (2001): The Attention Economy: Understanding the New Currency of Business, Boston, 2001.

Deemer, K. (1980): A Life Cycle Model of New Product Profitability. In: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small Systems, Palo Alto, 1980, S. 150-155.

Deitel, H. M.; Deitel, P. J. (1997): Java how to program, Upper Saddle River, 1997.

Dix, A. J. (2004): Human-Computer Interaction, 3. bearb. Aufl., London [u. a.], 2004.

Dornbusch, P.; Huber, M. (2003): Generierung von Ortsinformationen durch User-Communities. In: Uhr, W.; Esswein, W.; Schoop, E. (Hrsg.): Wirtschaftsinformatik 2003 – Medien, Märkte, Mobilität, Bd. 1, Heidelberg [u. a.], 2003, S. 175-186.

Dostal, W. et al. (2005): Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis, Heidelberg [u. a.], 2005.

Drucker, P. (2005): Was ist Management?: Das Beste aus 50 Jahren, 3. Aufl., Berlin, 2005.

ECMA (2005): Standard ECMA-335 Common Language Infrastructure (CLI), 3rd edition, http://www.ecma-international.org/publications/standards/Ecma-335.htm, Abgerufen am 11.04.06, 2005.

ECMA (2006): Standard ECMA-335 Common Language Infrastructure (CLI), 4th edition, http://www.ecma-international.org/publications/standards/Ecma-335.htm, Abgerufen am 20.12.06, 2006.

Eckert, C. (2004): IT-Sicherheit: Konzept – Verfahren – Protokolle, 3. überarb. u. erw. Aufl., München [u. a.], 2004.

Eicker, S.; Nietsch, M. (1999): Standards zum objektorientierten Paradigma. In: Wirtschaftsinformatik, Bd. 4, Nr. 5, 1999, S. 358-370.

Eicker, S.; Schwichtenberg, H.; Büning, W. (2001): Ein Framework für Shop-Systeme auf Basis der Java 2 Enterprise Edition. In: Buhl, H. U.; Huther, A.; Reitwiesner, B. (Hrsg.): Information Age Economy, Heidelberg, 2001, S. 637-650.

Eicker, S.; Schwichtenberg, H. (2003): Universal Component Trading – Trading in heterogenen Komponentenumgebungen. In: Uhr, W.; Esswein, W.; Schoop, E. (Hrsg.): Wirtschaftsinformatik 2003 – Medien, Märkte, Mobilität, Bd. 1, Heidelberg [u. a.], 2003, S. 303-323.

Eller, B. (2005): Klassifikation von Anwendungsgebieten im privaten und mobilen Bereich, Seminararbeit TU Darmstadt, Fachgebiet Wirtschaftsinformatik I, Darmstadt, 2005.

Elzenheimer, M.; Lonthoff, J.; Ortner, E. (2006): Requirements and Tools for Reasonable End User Development. In: Hochberger, C.; Liskowsky, R. (Hrsg.): INFORMATIK 2006 – Informatik für Menschen, LNI-Proceedings Bd. P-93 Nr. 1 zur 36. Jahrestagung der GI e.V., Bonn, 2006, S. 580-583.

Engel, A.; Koschel, A.; Tritsch, R. (2002): J2EE kompakt: Enterprise Java – Konzepte und Umfeld, Heidelberg [u. a.], 2002.

Ferstl, O. K.; Sinz, E. J. (2001): Grundlagen der Wirtschaftsinformatik, Bd. 1, 4. Aufl., München [u. a.], 2001.

Fettke, P.; Loos, P.; Viehweger, B. (2003): Komponentenmarktplätze – Bestandsaufnahme und Typologie. In: Turowski, K. (Hrsg.): Proceedings des 5. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 5), Augsburg, 2003, S. 1-15.

Fettke, P.; Loos, P.; von der Tann, M. (2002): Entwicklung eines Repositoriums für Fachkomponenten auf Grundlage des Vorschlages zur Vereinheitlichung der Spezifikation von Fachkomponenten – Analyse von Problemen und Diskussion von Lösungsalternativen. In: Turowski, K. (Hrsg.): Modellierung und Spezifikation von Fachkomponenten, Proceedings der Multikonferenz Wirtschaftsinformatik 2002 (MKWI02), Nürnberg, 2002, S. 87-117.

Fink, A.; Schneidereit, G.; Voß, S. (2001): Grundlagen der Wirtschaftsinformatik, Heidelberg, 2001.

Flake, S. (2003): Temporal OCL Extensions for Specification of Real-Time Constraints. In: Workshop Specification and Validation of UML models for Real Time and Embedded Systems (SVERTS'03) at UML 2003, San Francisco, 2003.

Fletcher, P.; Waterhouse, M. (2002): Web Services Business Strategies and Architectures, Indianapolis, 2002.

Frank, U. (2006): Towards a Pluralistic Conception of Research Methods in Information Systems Research, ICB-Research Report No. 7, Universität Duisburg-Essen, 2006.

Friedell, E. (1960): Kulturgeschichte der Neuzeit: die Krisis der europäischen Seele von der Schwarzen Pest bis zum Ersten Weltkrieg, Bd. 1 – Einleitung, Renaissance und Reformation, ungek. Ausg. in 1 Bd., 33. - 53 Tsd. Aufl., München, 1960.

Fritsch, D.; Klinec, D.; Volz, S. (2000): Nexus – Positioning and Data Management Concepts for Location Aware Applications. In: Proceedings of the 2nd International Symposium on Telegeoprocessing, Nice-Sophia-Antipolis, 2000, S. 171-184.

Galileo (2003): The Galilei Project – Galileo Design Consolidation, europa.eu.int/comm/dgs/energy_transport/galileo/doc/galilei_brochure.pdf., Abgerufen am 05.11.06, 2003.

Gamma, E. (2005): Design Patterns: Elements of reusable object-oriented Software, 32. Aufl., Boston [u. a.], 2005.

Garlan, D.; Shaw, M. (1993): An Introduction to Software Architecture. In: Ambriola, V.; Tortora, G. (Hrsg.): Advances in Software Engineering and Knowledge Engineering, Singapore, 1993, S. 1-39.

Gates, B. (1999): Remarks by Bill Gates Fusion '99, http://www.microsoft.com/billgates/speeches/07-23fusion.asp, Abgerufen am 07.04.06, 1999.

Gelernter, D. (1985): Generative Communication in Linda. In: ACM Transactions on Programming Languages and Systems, Bd. 7, Nr. 1, 1985, S. 80-112.

Geraghty, R. et al. (1999): COM-CORBA Interoperability, Upper Saddle River, 1999.

Gillenson, M. L. (1982): The State of Practice of Data Administration. In: Communications of the ACM, Bd. 25, Nr. 10, 1982, S. 699-706.

Goedicke, M.; Neumann, G.; Zdun, U. (2001): Design and Implementation Constructs for the Development of Flexible, Component-Oriented Software Architectures. In: Butler, G. (Hrsg.): LNCS-Proceedings of second international Symposium on Generative and Component based Software Engineering (GCSE 2000), Erfurt, Bd. 2177, Berlin [u. a.], 2001, S. 114-128.

Graeve, C. (2001): M-Commerce – Mobilität, Machbarkeit und Manie. In: HMD – Praxis der Wirtschaftsinformatik, Nr. 220, 2001, S. 5-14.

Griffel, F. (1998): Componentware: Konzepte und Techniken eines Softwareparadigmas, Heidelberg, 1998.

Grohmann, W. (2002): ASP Application Service Providing: Software auf Mietbasis: Kosten sparen. Vorteile nutzen, Köln, 2002.

Grollius, T. et al. (2005): An approach for the Management of Service-oriented Architecture (SoA) based Application Systems. In: Desel, J.; Frank, U. (Hrsg.): LNI-Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures (EMISA'05), Klagenfurt, Bd. P-75, Bonn, 2005, S. 208-221.

Grollius, T.; Lonthoff, J.; Ortner, E. (2006a): Externes Anwendungsmanagement für ubiquitäre Anwendungssysteme. In: Lehner, F.; Nösekabel, H.; Kleinschmidt, P. (Hrsg.): Tagungsband zur Multikonferenz Wirtschaftsinformatik 2006 (MKWI 2006), Bd. 2, Berlin, 2006, S. 363-373.

Grollius, T.; Lonthoff, J.; Ortner, E. (2006b): An Approach to Manage Flexible Component-Based Application Systems. In: Isaías, P.; McPherson, M.; Bannister, F. (Hrsg.): Proceedings of IADIS International Conference e-Society 2006, Bd. 2, Dublin, 2006, S. 219-223.

Grollius, T. (2004): Benutzungsplanung und -steuerung komponentenbasierter Anwendungssysteme, Diplomarbeit TU Darmstadt Fachgebiet Wirtschaftsinformatik I, Darmstadt, 2004.

Gruhn, V.; Thiel, A. (2000): Komponentenmodelle: DCOM, JavaBeans, Enterprise-JavaBeans, CORBA, München [u. a.], 2000.

Hanhart, D.; Legner, C.; Österle, H. (2005): Anwendungsszenarien des Mobile und Ubiquitous Computing in der Instandhaltung. In: Hampe, F. et al. (Hrsg.): Mobile Business – Processes, Platforms, Payments – LNI-Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), Bd. P-59, Bonn, 2005, S. 45-58.

Hansen, H. R.; Neumann, G. (2001): Wirtschaftsinformatik I, 8. Aufl., Stuttgart, 2001.

Hansen, H. R.; Neumann, G. (2005): Wirtschaftsinformatik 1: Grundlagen und Anwendungen, 9. Aufl., Stuttgart, 2005.

Hanser, P. (1995): Aufbruch in den Cyberspace. In: asw –absatzwirtschaft, Bd. 38, Nr. 8, 1995, S. 34-39.

Hansmann, U. et al. (2001): Pervasive Computing Handbook, Berlin [u. a.], 2001.

Harmon, P. (2000): What Types of Components are Companies using, http://www.cutter.com/research/2000/crb001219.html, Abgerufen am 11.04.06, 2000.

Hasselbring, W. (2006): Software-Architektur. In: Informatik Spektrum, Bd. 29, Nr. 1, 2006, S. 48-52.

Hasselmeyer, P.; Kehr, R.; Voß, M. (2000): Trade-offs in a Secure Jini Service Architecture. In: Linnhoff-Popien, C.; Hegering, H.-G. (Hrsg.): LNCS-Proceedings of 3rd IFIP/GI International Conference on Trends towards a Universal Service Market (USM 2000), Bd. 1890, Berlin [u. a.], 2000, S. 190-201.

Hau, M.; Mertens, P. (2002): Computergestützte Auswahl komponentenbasierter Anwendungssysteme. In: Informatik-Spektrum, Bd. 25, Nr. 5, 2002, S. 331 - 340.

Hautzinger, H.; Tassaux-Becker, B.; Pfeiffer, M. (1996): Gesetzmäßigkeiten des Mobilitätsverhaltens – Verkehrsmobilität in Deutschland zu Beginn der 90er Jahre. In: Bundesanstalt für Straßenwesen (Hrsg.): Berichte, Bd. 4, Heft M57, Bergisch-Gladbach, 1996.

Hayes-Roth, B. et al. (1995): A Domain-Specific Software Architecture for Adaptive Intelligent Systems. In: IEEE Transactions on Software Engineering, Bd. 21, Nr. 4, 1995, S. 288-301.

Heinemann, E. (2006): Sprachlogische Aspekte rekonstruierten Denkens, Redens und Handelns: Aufbau einer Wissenschaftstheorie der Wirtschaftsinformatik, Dissertation TU Darmstadt, Fachgebiet Wirtschaftsinformatik I, Wiesbaden, 2006.

Heinrich, L. J. (2002): Wirtschaftsinformatik, Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur, 7. Aufl., München [u. a.], 2002.

Hellmuth, T. W. (1997): Terminologiemanagement in der Informationsverarbeitung: Aspekte einer effizienten Kommunikation in der computerunterstützten Informationsverarbeitung, Dissertation Universität Konstanz, Berlin, 1997.

Henderson-Sellers, B.; Edwards, J. M. (1994): The working Object: Object-oriented Software Engineering – Methods and Management – Book two of Object-oriented Knowledge, Sydney [u. a.], 1994.

Hilty, L.; et al. (2003): Das Vorsorgeprinzip in der Informationsgesellschaft, Auswirkungen des Pervasive Computing auf Gesundheit und Umwelt: Studie des Zentrums für Technologie-Abschätzung, Nr. 46, Bern, 2003.

Hitz, M. et al. (2005): UML@Work: objektorientierte Modellierung mit UML2, 3. akt. u. überarb. Aufl., Heidelberg, 2005.

Holzapfel, H. (2001): Drang nach Überallität. In: fairkehr, Nr.6, 2001.

Häckelmann, H.; Petzold, H.-J.; Strahringer, S. (2000): Kommunikationssysteme: Technik und Anwendungen, Berlin [u. a.], 2000.

Härder, T.; Rahm, E. (1999): Datenbanksysteme: Konzepte und Techniken der Implementierung, Berlin [u. a.], 1999.

Höpfner, H.; Türker, C.; König-Ries, B. (2005): Mobile Datenbanken und Informationssysteme: Konzepte und Techniken, Heidelberg, 2005.

Höß, O. (2005): Ein System für das Wiederverwendungs-Management von Software-Komponenten, Dissertation Universität Stuttgart, Fakultät Maschinenbau, Stuttgart, 2005.

Hümmer, W. (2004): Vertragsverhandlungen um konfigurierbare Produkte im elektronischen Handel, Dissertation Friedrich-Alexander-Universität Erlangen-Nürnberg, Fakultät Informatik 6, Erlangen, 2004.

IBM (o. J.): Autonomic Computing, http://www.research.ibm.com/autonomic/overview/, Abgerufen am 02.10.06, o. J.

IEEE (1990a): IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, New York, 1990.

IEEE (1990b): IEEE standard glossary of software engineering terminology, http://www.ieee/iel1/2238/4148/00159342.pdf?tp=&arnumber=159342&isnumber=4148, Abgerufen am 05.11.06, 1990.

IEEE (2000): IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Standard 1471-2000, 2000.

IEEE (2006a): IEEE 802.15 WPAN Task Group 1 (TG1), http://www.ieee802.org/15/pub/ TG1.html, Abgerufen am 08.12.06, 2006.

IEEE (2006b): The IEEE 802.16 Working Group on Broadband Wireless Access Standards, http://grouper.ieee.org/groups/802/16/, Abgerufen am 13.12.06, 2006.

IETF; Dierks, T.; Rescorla, E. (2006): The Transport Layer Security (TLS) Protocol Version 1.1, http://www.ietf.org/rfc/rfc4346.txt, Abgerufen am 22.11.06, 2006.

IETF; Postel, J. B.; Reynolds, J. (1985): File Transfer Protocol, http://www.ietf.org/rfc/rfc959, Abgerufen am 22.11.06, 1985.

IETF; Postel, J. B. (1982): Simple Mail Transfer Protocol, http://www.ietf.org/rfc/rfc0821.txt, Abgerufen am 22.11.06, 1982.

IETF; Srinivasan, R. (1995): RPC: Remote Procedure Call Protocol Specification Version 2, http://tools.ietf.org/html/rfc1831, Abgerufen am 22.11.06, 1995.

ISO (2005): Common Language Infrastructure, ISO-Dokument Nr. ISO/IEC 23271:2003, Abgerufen am 11.04.06, 2005.

ISO (1996): Extended BNF - ISO/IEC 14977:1996, http://www.iso.ch/iso/en/ CatalogueDetailPage.CatalogueDetail?CSNUMBER=26153, Abgerufen am 22.11.06, 1996.

ISO (1998): ISO 9241 Ergonomics of human-system interaction - Part 10: Dialogue principles, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38009&ICS1=13&ICS2=180&ICS3=, Abgerufen am 21.12.05, 1998.

ISO (2006): International Organization for Standardization, http://www.iso.org, Abgerufen am 22.11.06, 2006.

Infrared Data Association (2006): IrDA, www.irda.org, Abgerufen am 11.12.06, 2006.

Inmarsat (o. J.): Total Communications Network, www.inmarsat.org, o. J.

Jablonski, S. (1997): Workflow-Management: Entwicklung von Anwendungen und Systemen, Heidelberg, 1997.

Jablonski, S. (2004): Guide to web application and platform architectures, Berlin [u. a.], 2004.

José, D.; Davis, N. (1999): Scalable and Flexible Location-Based Services for Ubiquitous Information Access. In: LNCS-Proceedings of First International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Bd. 1707, 1999, S. 52-66.

Kamlah, W.; Lorenzen, P. (1996): Logische Propädeutik: Vorschule des vernünftigen Redens, 3. Aufl., Stuttgart, 1996.

Kehr, R. (2000): Spontane Vernetzung. In: Informatik Spektrum, Bd. 23, Nr. 3, 2000, S. 161-172.

Kephart, J. O.; Chess, D. M. (2003): The Vision of Autonomic Computing. In: IEEE Computer Magazine, Bd. 36, Nr. 1, 2003, S. 41-50.

Kim, G. (2005): Designing Virtual Reality Systems: The structured Approach, London [u. a.], 2005.

Kleinrock, L. (1996): Nomadicity: anytime, anywhere in a disconnected World. In: Mobile Network Applications, Bd. 1, Nr. 4, 1996, S. 351-357.

Kollmann, T. (1998): The Information Triple Jump as the Measure of Success in Electronic Commerce. In: EM - Electronic Markets, Bd. 8, Nr. 4, 1998, S. 44-49.

Kollmann, T. (2001): Virtuelle Marktplätze: Grundlagen – Management – Fallstudien, München, 2001.

Kosiol, E. (1972): Die Unternehmung als wirtschaftliches Aktionszentrum: Einführung in die Betriebswirtschaftslehre, Reinbeck bei Hamburg, 1972.

Koster, K. (2002): Ein Streifzug durch die Welt der mobilen Endgeräte. In: Hartmann, D. (Hrsg.): Business Computing – Geschäftsprozesse mit Mobile Computing: konkrete Projekterfahrung, technische Umsetzung, kalkulierbarer Erfolg des Mobile Business, Braunschweig [u. a.], 2002, S. 127-145.

Krafzig, D.; Banke, K.; Slama, D. (2005): Enterprise SOA: Service-Oriented Architecture Best Practices, Upper Saddle River, 2005.

Krcmar, H. (2000): Informationsmanagement, 2. Aufl., Berlin [u. a.], 2000.

Kruchten, P. (2004): The Rational Unified Process, 2. Aufl., Reading [u. a.], 2004.

Kruse, W. (2003): Lebenslanges Lernen in Deutschland – Finanzierung und Innovation: Kompetenzentwicklung, Bildungsnetze, Unterstützungsstrukturen. In: BMBF (Hrsg.): Bericht des BMBF für die OECD zu ”Good Practice der Finanzierung Lebenslangen Lernens” im Rahmen des Projektes Co-financing lifelong learning, 2003.

Kuhn, J. (2003): Kommerzielle Nutzung mobiler Anwendungen, Dissertation Universität Regensburg, Wirtschaftswissenschaftliche Fakultät, Regensburg, 2003.

Kuhn, T. S. (1967): Die Struktur wissenschaftlicher Revolutionen, Frankfurt a. M., 1967.

Ledoux, T. (1999): OpenCorba: a Reflective Open Broker. In: LNCS-Proceedings of the Second International Conference Reflection'99, Saint-Malo, Bd. 1616, 1999, S. 197-214.

Lehmann, F. R. (1998): Aktuelles Schlagwort: Normsprache. In: Informatik Spektrum, Bd. 21, Nr. 6, 1998, S. 366-367.

Lehner, F. (1989): Nutzung und Wartung von Software. Das Anwendungssystem-Management, München, 1989.

Lehner, F. (2003): Mobile und drahtlose Informationssysteme: Technologien, Anwendungen, Märkte, Berlin [u. a.], 2003.

Lehner, F. (1966): Wechselbeziehungen zwischen Städtebau und Nahverkehr. In: Schriftenreihe für Verkehr und Technik, Nr. 29, 1966.

Lehner, W. (2002): Subskriptionssysteme: Marktplatz für omnipräsente Informationen, Stuttgart [u. a.], 2002.

Leinweber, G.; Andresen, T. (1992): Software- und Anwendungsmanagement: Strategien und Entscheidungshilfen für die Unternehmenspraxis, München [u. a.], 1992.

Lipp, L. (2004): Interaktion zwischen Mensch und Computer im Ubiquitos Computing: Alternative Ein- und Ausgabemöglichkeiten für allgegenwärtige Informationstechnologien, Münster, 2004.

Longshaw, A. (2001): Choosing Between COM+, EJB, and CCM. In: Heinemann, G. T.; Councill, W. T. (Hrsg.): Component-based Software Engineering, Upper Saddle River, 2001, S. 621-640.

Lonthoff, J.; Leiber, T. (2005): Mobile Location Based Gaming als Wegbereiter für Location Based Services. In: von Knop, J.; Haverkamp, W.; Jessen, E. (Hrsg.): "Heute schon das Morgen sehen", LNI-Proceedings zur 19. DFN-Arbeitstagung über Kommunikationsnetze, Düsseldorf, Bd. P-73, 2005, S. 343-360.

Lonthoff, J.; Ortner, E., Schmidt-Eisenlohr, K. (2006a): Mobile Anwendungssysteme im privaten und beruflichen Bereich. In: Hochberger, C.; Liskowsky, R. (Hrsg.): INFORMATIK 2006 – Informatik für Menschen, LNI-Proceedings zur 36. Jahrestagung der GI e.V., Bonn, Bd. P-93, Nr. 1, 2006, S. 187-188.

Lonthoff, J.; Ortner, E. (2006b): Mobile Location-Based Gaming as Enabler for Location-Based Services (LBS). In: Isaías, P.; McPherson, M.; Bannister, F. (Hrsg.): Proceedings of IADIS International Conference e-Society 2006, Bd. 1, Dublin, Ireland, 2006, S. 485-492.

Lonthoff, J.; Ortner, E. (2007): Klassifikations- und Lösungsansätze für Web Services im mobilen Umfeld. In: LNI-Proceedings der 2. Konferenz Mobilität und Mobile Informationssysteme (MMS 2007), Aachen, Bd. P-104, 2007, S. 73-84.

Lonthoff, J.; Wolf, M., Hoffmann, M. (2006): Mobile Couponing – Coupon- und Rabattsysteme für Location Based Services. In: Hochberger, C.; Liskowsky, R. (Hrsg.): INFORMATIK 2006 – Informatik für Menschen, LNI-Proceedings zur 36. Jahrestagung der GI e.V., Bonn, Bd. P-93, Nr. 1, 2006, S. 248-254.

Lonthoff, J. (2004): Projekt mobiCOMP. In: von Knop, J.; Haverkamp, W.; Jessen, E. (Hrsg.): E-Science und Grid – Ad-hoc-Netze – Medienintegration, LNI-Proceedings zur 18. DFN-Arbeitstagung über Kommunikationsnetze, Bonn, Bd. P-55, 2004, S. 451-465.

Lonthoff, J. (2007): Enterprise Mobility. In: T-Systems (Hrsg.): White Paper – Enterprise Mobility: Durch Mobilität zum Erfolg, Frankfurt, 2007.

Lorenz, K. (1992): Einführung in die philosophische Anthropologie, Darmstadt, 1992.

Lorenzen, P. (1973): Semantisch normierte Orthosprachen. In: Kambartel, F.; Mittelstraß, J. (Hrsg.): Zum normativen Fundament der Wissenschaft, Frankfurt: Athenäum, 1973, S. 231-249.

Lorenzen, P. (1985): Grundbegriffe technischer und politischer Kultur, Frankfurt, 1985.

Malkewitz, R. (2004): Ambient Intelligence. Stand und Trends, http://www.zgdv.de/zgdv/zgdv20/Vortraege/STAR-AmI-abstract, Abgerufen am 30.04.06, 2004.

Mascolo, C. et al. (2002): XMIDDLE: A Data-Sharing Middleware for Mobile Computing. In: International Journal on Personal and Wireless Communications, Bd. 1, Nr. 21, 2002, S. 77-103.

Mattern, F.; Fleisch, E. (2005): Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis, Berlin [u. a.], 2005.

Mattern, F. (1999): Infrastruktur für den elektronischen Markt. In: Praxis der Informationsverarbeitung und Kommunikation, Bd. 22, Nr. 3, 1999, S. 185.

Mattern, F. (2003): Total vernetzt: Szenarien einer informatisierten Welt, Berlin [u. a.], 2003.

McGregor, J. D.; Sykes, D. A. (1992): Object-Oriented Software Development: Engineering Software for Reuse, New York, 1992.

Meeks, M. (2001): Bonobo and Free Software GNOME Components. In: Heinemann, G. T.; Councill, W. T. (Hrsg.): Component-based Software Engineering, Upper Saddle River, 2001, S. 607-619.

Meffert, H. (2000): Marketing: Grundlagen marktorientierter Unternehmensführung – Konzepte, Instrumente, Praxisbeispiele, 9. überarb. u. erw. Aufl., Wiesbaden, 2000.

Mertens, P. (2001): Lexikon der Wirtschaftsinformatik, 4. vollst. neu bearb. u. erw. Aufl., Berlin [u. a.], 2001.

Mertens, P. (2005a): Grundzüge der Wirtschaftsinformatik, 9. Aufl., Berlin [u. a.], 2005.

Mertens, P. (2005b): Wirtschaftsinformatik – Aktuelle Probleme eines interdisziplinären Faches, http://ahr.informatik.uni-kl.de/pubs/Dagstuhl/Mertens.dbag-2005.pdf, Abgerufen am 01.07.05, 2005.

Merz, M.; Tu, T.; Lamersdorf, W. (1999): Electronic Commerce. In: Informatik Spektrum, Bd. 22, Nr. 5, 1999, S. 328-343.

Merz, M. (2002): E-commerce und e-business: Marktmodelle, Anwendungen und Technologien, 2. Aufl., Heidelberg, 2002.

Microsoft (o. J.): COM: Componet Object Model Technologies, http://www.microsoft.com/com/default.mspx, Abgerufen am 11.04.06, o. J.

Microsoft (2006a): Microsoft Windows Mobile, www.microsoft.com/windowsmobile/, Abgerufen am 08.12.06, 2006.

Microsoft (2006b): .NET Compact Framework, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_evtuv/html/etconNETCompactFramework.asp, Abgerufen am 08.12.06, 2006.

Microsoft (2006c): Managing Mobile Devices in the Enterprise, http://www.microsoft.com/technet/itsolutions/mobile/evaluate/mblmange.mspx, Abgerufen am 08.12.06, 2006.

Microsoft (2006d): Systems Management Server, http://www.microsoft.com/smserver/default.mspx, Abgerufen am 08.12.06, 2006.

Microsoft (2006e): Mobile Information Server, http://www.microsoft.com/technet/prodtechnol/mis, Abgerufen am 08.12.06, 2006.

mik21 (2004): mik21 Projektvorhaben. Migrationskompetenz als Schlüsselfaktor der Ökonomie des 21. Jahrhunderts, http://www.mik21.uni-kassel.de/download/mik21-Projektvorhaben.pdf, Abgerufen am 28.04.06, 2004.

Mili, H.; Mili, F.; Mili, A. (1995): Reusing Software – Issues and Research Directions. In: IEEE Transactions on Software Engineering, Bd. 21, Nr. 6, 1995, S. 529-561.

Mili, H. et al. (2002): Reuse-Based Software Engineering – Techniques, Organization, and Controls, New York, 2002.

Mittelstraß, J. (2004): Enzyklopädie Philosophie und Wissenschaftstheorie, Bd. 2: H-O, Stuttgart [u. a.], 2004.

Modahl, M. et al. (2006): UbiqStack: a Taxonomy for a Ubiquitous Computing Software Stack. In: Personal and Ubiquitous Computing, Bd. 10, Nr. 1, 2006, S. 21-27.

Mutschler, B.; Specht, G. (2004): Mobile Datenbanksysteme: Architektur, Implementierung, Konzepte, Berlin [u. a.], 2004.

Müller-Wilken, S. (2002): Mobile Geräte in verteilten Anwendungsumgebungen: Ein Integrationsansatz zwischen Abstraktion und Migration, Dissertation Universität Hamburg, 2002.

Nielsen, J. (2003): Usability engineering, Amsterdam [u. a.], 2003.

Niemann, F. (2006): SOA zwischen Mythos und Realität. In: Computerwoche Nr. 5, 2006, S. 6.

Noam. E. M. (1997): Systematic Bottlenecks in the Information Society. In: European Communication Council (ECC) - Report 1997, Berlin, 1997, S. 35-44.

OASIS (2006): OASIS UDDI Specification TC, http://www.oasis-open.org/ committees/tc_home.php?wg_abbrev=uddi-spec, Abgerufen am 22.11.06, 2006.

OMA (1998): WMLScript Specification, http://www.wapforum.org/what/technical/ wmlss-30-apr-98.pdf, Abgerufen am 23.11.06, 1998.

OMA (2001a): Wireless Markup Language Version 2.0, http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-238-wml-20010911-a.pdf, Abgerufen am 22.11.06, 2001.

OMA (2001b): WAP Architecture, http://www.openmobilealliance.org/tech/affiliates/ LicenseAgreement.asp?DocName=/wap/wap-210-waparch-20010712-a.pdf, Abgerufen am 22.11.06, 2001.

OMA (2001c): Wireless Transaction Protocol, http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-224-wtp-20010710-a.pdf, Abgerufen am 22.11.06, 2001.

OMA (2001d): Wireless Transport Layer Security, http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-261-wtls-20010406-a.pdf, Abgerufen am 22.11.06, 2001.

OMG (1996): Common Facilities RFP-5 (Meta-Object Facility), OMG TC Document cf/96-02-01, Revision 2, 1996.

OMG (2004): Common Object Request Broker Architecture: Core Specification Version 3.0.3, http://www.omg.org/cgi-bin/doc?formal/04-03-01, Abgerufen am 22.11.06, 2004.

OMG (2006): CORBA Component Model Specification, v4.0, http://www.omg.org/cgi-bin/doc?formal/06-04-01, Abgerufen am 11.04.06, 2006.

OMG (2002): OMG IDL Syntax and Semantics, http://www.omg.org/cgi-bin/apps/doc?formal/02-06-39.pdf, Abgerufen am 04.05.06, 2002.

OMG (2003): Response to the UML 2.0 OCL RfP (ad/2000-09-03), http://www.omg.org/cgi-bin/doc?ad/03-01-07.pdf, Abgerufen am 06.05.06, 2003.

OMG (2006): The Object Management Group (OMG), http://www.omg.org/, Abgerufen am 22.11.06, 2006.

Oestereich, B. (2005): Analyse und Design mit UML 2: objektorientierte Softwareentwicklung, 7. akt. Aufl., München [u. a.], 2005.

Opennet-Initiative (2006): Opennet, wiki.opennet-initiative.de, Abgerufen am 13.12.06, 2006.

Ortner, E.; Overhage, S. (2003a): E-NOgS: Ein komponentenorientiertes Middleware-Framework für E-Commerce-Anwendungen. In: Thema Forschung, Nr. 1, 2003, S. 22-27.

Ortner, E.; Overhage, S. (2003b): CompoNex: Ein elektronischer Marktplatz für den Handel mit Software-Komponenten über das Internet. In: Thema Forschung, Nr. 1, 2003, S. 28-32.

Ortner, E.; Söllner, B. (1989): Semantische Datenmodellierung nach der Objekttypenmethode. In: Informatik Spektrum, Bd. 12, 1989, S. 31-42.

Ortner, E.; Söllner, B. (1987): Management betrieblicher Informationssysteme mit Data Dictionary-Unterstützung. In: Strohl-Goebel, H. (Hrsg.): Deutscher Dokumentartag 1987 – Von der Information zum Wissen, vom Wissen zur Information: Traditionelle und moderne Informationssysteme für Wissenschaft und Praxis, Weinheim, 1987, S. 127-150.

Ortner, E.; Wedekind, H. (2003): Die Zukunft des Bürgers im Internet. In: Arbeitsbericht Nr. 2, Darmstadt, 2003.

Ortner, E. (1983): Rekonstruktion von Begriffen und Begriffsbeziehungen beim Datenbankentwurf. In: Dahlberg, I.; Schader, M. R. (Hrsg.): Studien zur Klassifikation – Automatisierung in der Klassifikation, Bd. 13, Frankfurt, 1983, S. 76-88.

Ortner, E. (1984a): Modellierung von Datenbankanwendungen – Datenbankentwurf auf der Ebene von Benutzerfachsprachen. In: HMD – Praxis der Wirtschaftsinformatik, Nr. 118, 1984, S. 37-49.

Ortner, E. (1984b): Datenbankentwurf auf der Ebene der Benutzer - dargestellt am Beispiel von Buchungs- und Abrechnungssystemen. In: Schiemenz, B.; Wagner, A. (Hrsg.): Angewandte Wirtschafts- und Sozialkybernetik: neue Ansätze in Praxis und Wissenschaft, Bd. 51, Berlin, 1984, S. 167-181.

Ortner, E. (1991a): Informationsmanagement: Wie es entstand, was es ist und wohin es sich entwickelt. In: Informatik Spektrum, Bd. 14, 1991, S. 315-327.

Ortner, E. (1991b): Ein Referenzmodell für den Einsatz von Dictionary-/ Repository-Systemen in Unternehmen. In: Wirtschaftsinformatik, Bd. 33, Nr. 5, 1991, S. 420-430.

Ortner, E. (1993): Software-Engineering als Sprachkritik: die sprachkritische Methode des fachlichen Software-Entwurfs, Konstanz, 1993.

Ortner, E. (1999): Repository Systems. Teil 1: Mehrstufigkeit und Entwicklungsumgebung. In: Informatik-Spektrum, Bd. 22, Nr. 4, 1999, S. 235 - 251.

Ortner, E. (2002): Sprachingenieurwesen: Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik. In: Informatik Spektrum, Bd. 25, Nr. 1, 2002, S. 39-51.

Ortner, E. (2003): Vom Wort zum Bauelement. In: Thema Forschung, Nr. 1, 2003.

Ortner, E. (2004): Anthropozentrik und Sprachbasierung in der (Wirtschafts-)informatik. In: Hammwöhner, R.; Rittberger, M.; Semar, W. (Hrsg.): Wissen in Aktion – Der Primat der Pragmatik als Motto der Konstanzer Informationswissenschaft, Bd. 41, Konstanz, 2004, S. 141-152.

Ortner, E. (2005a): Sprachbasierte Informatik: Wie man mit Wörtern die Cyber-Welt bewegt, Leipzig, 2005.

Ortner, E. (2005b): Forschungsprogramm 2005, Darmstadt, 2005.

Ortner, E. (2006): Entwicklung von Anwendungssystemen II, Vorlesungsfolien, Darmstadt, 2006.

Overhage, S. (2002): Die Spezifikation – kritischer Erfolgsfaktor der Komponentenorientierung. In: Turowski, K. (Hrsg.): Proceedings des 4. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Augsburg, 2002, S. 1-18.

Overhage, S. (2006): Vereinheitlichte Spezifikation von Komponenten: Grundlagen, UnSCom Spezifikationsrahmen und Anwendung, Dissertation Universität Augsburg, Fakultät Wirtschaftsinformatik und Systems Engineering, Augsburg, 2006.

Pagé P. (1992): Strategien in der Software-Produktion. In: Leinweber, G. (Hrsg.): Software- und Anwendungsmanagement, München, 1992, S. 43-104.

Pandya, R. (2000): Mobile and personal Communication Systems and Services, Piscataway, 2000.

Parnas, D. L. (1972): On the Criteria to be used in Decomposing Systems into Modules. In: Communications of the ACM, Bd. 15, Nr. 12, 1972, S. 1053-1058.

Patrick, P. (2005): Impact of SOA on Enterprise Information Architectures. In: Proceedings of the 2005 ACM SIGMOD international Conference on Management of Data, New York, 2005, S. 844-848.

Perry, D. E.; Wolf, A. L. (1992): Foundations for the Study of Software Architecture. In: ACM Sigsoft Software Engineering Notes, Bd. 17, Nr. 4, 1992, S. 40-52.

Picco, G. P.; Murphy, A. L.; Roman, G.-C. (1999): LIME: Linda meets mobility. In: IEEE Computer Society Press (Hrsg.): Proceedings of the 21st international conference on software engineering (ICSE'99), Los Angeles, 1999, S. 368-377.

Preuß, S. (2000): NetObjects – Dynamische Proxy-Architektur für Jini. In: Proceedings of Net.ObjectDays 2000, Erfurt, 2000, S. 146-155.

Prieto-Diaz, R. (1993): Status Report: Software Reusability. In: IEEE Software, Bd. 10, Nr. 3, 1993, S. 61-66.

Ramakrishnan, S.; McGregor, J. (1999): Extending OCL to Support Temporal Operators. In: Proceedings of 21st International Conference on Software Engineering (ICSE99), Workshop on Testing Distributed Component-Based Systems, L.A., 1999.

Ravichandran, T.; Rothenberger, M. (2003): Software Reuse Strategies and Component Markets. In: Communications of the ACM, Bd. 46, Nr. 8, 2003, S. 109-114.

Reiche, D. (2003): Roche-Lexikon Medizin, 5. Aufl., München [u. a.], 2003.

Repp, N. (2003): mobileISM – Entwicklung eines Frameworks zur benutzerunterstützenden Verwaltung von Diensten auf mobilen Endgeräten. In: Wirtschaftsinformatik, Bd. 45, Nr. 6, 2003, S. 658-661.

Reussner, R.; Hasselbring, W. (2006): Handbuch der Software-Architektur, Heidelberg, 2006.

Ridder, L. (2002): Mereologie: ein Beitrag zur Ontologie und Erkenntnistheorie. In: Philosophische Abhandlungen, Bd. 83, Frankfurt, 2002.

Rischpater, R. (2000): WAP und WML: Wireless Web – Das neue Internet, Bonn, 2000.

Rorty, R. M. (1967): The linguistic Turn: Essays in Philosophical Method, Chicago [u. a.], 1967.

Rotach, M.; Hoppler, F.; Mötteli, M (1986): Erarbeitung eines Forschungsprogrammes Mobilität. Forschungsauftrag ETH Zürich, Institut für Verkehrsplanung, Transporttechnik, Straßen- und Eisenbahnbau, Zürich, 1986.

Roth, J. (2002): Mobile Computing: Grundlagen, Technik, Konzepte, Heidelberg, 2002.

Roth, J. (2005): Mobile Computing: Grundlagen, Technik, Konzepte, 2. akt. Aufl., Heidelberg, 2005.

Rothermel, K. (2005): SFB 627: Nexus Umgebungsmodelle für Mobile Kontextbezogene Systeme, http://www.nexus.uni-stuttgart.de/, Abgerufen am 22.11.06, 2005.

Royce, W. (1970): Managing the Development of Large Software Systems: Concepts and Techniques. In: Proceedings of IEEE WESCON, Washington, 1970, S. 1-9.

Röttger-G., S (2002): Lokalisierungsmethoden. In: Gora, W.; Röttger-Gerigk, S. (Hrsg.): Handbuch Mobile-Commerce: technische Grundlagen, Marktchancen und Einsatzmöglichkeiten, Berlin [u. a.], 2002, S. 419-426.

Rürup, B. (1998): Wohlfahrtsstaatliche Politik in der globalisierten Informationsgesellschaft. In: Schriften der Friedrich-Ebert-Stiftung, Bonn, 1998.

Sametinger, J. (1997): Software Engineering with reusable Components, Berlin [u. a.], 1997.

Sang-Bum Park, A. (2004): A Service-Based Agent System Supporting Mobile Computing, Dissertation RWTH Aachen, Aachen, 2004.

Sarkar, M. B.; Butler, B.; Steinfield, C. (1995): Intermediaries and Cybermediaries: A Continuing Role for Mediating Players in the Electronic Marketplace. In: Journal of Computer-Mediated Communication, Bd. 1, Nr. 3, 1995.

Satyanarayanan, M. et al. (1990): Coda: a highly available file system for a distributed workstation environment. In: IEEE Transactions on Computers, Bd. 39, Nr. 4, 1990, S. 447-459.

Scheidl, S. (2006): Workshop End-User Development. In: Hochberger, C.; Liskowsky, R. (Hrsg.): INFORMATIK 2006 – Informatik für Menschen, Bd. 1, Bonn, 2006, S. 563.

Schiemann, H. (1992): Glossar. In: Leinweber, G. (Hrsg.): Software- und Anwendungsmanagement, München, 1992, S. 253-265.

Schienmann, B. (1997): Objektorientierter Fachentwurf, Stuttgart, 1997.

Schiller, J. (2000): Mobilkommunikation: Techniken für das allgegenwärtige Internet, London [u. a.], 2000.

Schiller, J. (2003): Mobilkommunikation, 2. überarb. Aufl., München [u. a.], 2003.

Schmid, B. (1993): Elektronische Märkte. In: Wirtschaftsinformatik, Bd. 35, Nr. 5, 1993, S. 465-480.

Schneider, H. J. (1992): Phantasie und Kalkül: über die Polarität von Handlung und Struktur in der Sprache, Frankfurt, 1992.

Schulze, W. (1999): Ein Workflow-Management-Dienst für ein verteiltes Objektverwaltungssystem, Dissertation TU Dresden, Fakultät Informatik, Dresden, 1999.

Schwanke, R. W.; Altucher, R. Z.; Platoff, M. A. (1989): Discovering, Visualizing, and Controlling Software Structure. In: Proceedings of 5th International Workshop on Software Specification and Design IWSSD, Pittsburgh, 1989, S. 147-150.

Schwichtenberg, H. (2003): Universal Component Trading, Dissertation Universität Duisburg-Essen, Fakultät Wirtschaftswissenschaften, Essen, 2003.

Senko, M. E. et al. (1972): Specifications in a Data Independent Accessing Model. In: Proceedings of 1972 ACM-SIGFIDET workshop on Data description, access and control, New York, 1972, S. 363-382.

Shannon, C. E.; Weaver, W. (1949): The Mathematical Theory of Communication, Urbana, 1949.

Shaw, M. (1989): Larger Scale Systems Require Higher-Level Abstractions. In: Proceedings of 5th International Workshop on Software Specification and Design IWSSD, Pittsburgh, 1989, S. 143-146.

Siedersleben, J. (2003): Softwaretechnik, 2. überarb. u. akt. Aufl., München [u. a.], 2003.

SEI (2006): How Do You Define Software Architecture? Software Engineering Institute (SEI), Carnegie Mellon University, www.sei.cmu.edu/architecture/definitions.html, Abgerufen am 27.02.06, 2006.

Stahlknecht, P.; Hasenkamp, U. (2005): Einführung in die Wirtschaftsinformatik, 11. Aufl., Berlin [u. a.], 2005.

Stevens, P.; Pooley, R. J. (2000): Using UML: Software Engineering with Objects and Components, akt. Aufl., Harlow [u. a.], 2000.

Stowasser, A. C.; Petschenig, J. M.; Skutsch, M. F. (1994): Lateinisch-deutsches Schulwörterbuch, Wien, 1994.

Sun (o. J.): Java Platform, Enterprise Edition (Java EE), http://java.sun.com/javaee/index.jsp, Abgerufen am 11.04.06, o. J.

Sun (o. J.): Java 2 Platform, Micro Edition, http://java.sun.com/j2me/j2me-ds.pdf, Abgerufen am 22.11.06, o. J.

Sun (2006a): Jini Network Technology, sun.com/jini/, Abgerufen am 22.11.06, 2006.

Sun (2006b): Remote Method Invocation Home, http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp, Abgerufen am 22.11.06, 2006.

Sun (2006c): The Java ME Platform, http://java.sun.com/javame/index.jsp, Abgerufen am 22.11.06, 2006.

Sun (2006d): Mobile Information Device Profile (MIDP), http://java.sun.com/products/midp/, Abgerufen am 22.11.06, 2006.

Sun (2006e): Connected Limited Device Configuration (CLDC), http://java.sun.com/products/cldc/, Abgerufen am 22.11.06, 2006.

Szyperski, C.; Pfister, C. (1997): Workshop on Component-Oriented Programming Summary. In: Mühlhäuser, M. (Hrsg.): Special issues in Object Oriented Programming – Workshop Reader of the 10th European Conference on Object Oriented Programming ECOOP '96, Linz, 1997, S. 127-130.

Szyperski, C.; Gruntz, D. W.; Murer, S. (2002): Component software: beyond object-oriented programming, 2. Aufl., London [u. a.], 2002.

Szyperski, C. (1998): Component software: beyond object-oriented programming, Harlow [u. a.], 1998.

Tandler, P. (2004): Synchronous Collaboration in Ubiquitous Computing Environments, Dissertation TU Darmstadt, Fakultät Informatik, Darmstadt, 2004.

Tanenbaum, A. S. (1998): Computernetzwerke, 3. Aufl., München [u. a.], 1998.

Tanenbaum, A. S. (2003): Computernetzwerke, 4. überarb. Aufl., München [u. a.], 2003.

Tarski, A. (1979): Der Wahrheitsbegriff in den formalisierten Sprachen. In: Berka, K.; Kreiger, L. (Hrsg.): Logik Texte, Berlin, 1979.

Taubner, D. (2005): Software-Industrialisierung. In: Informatik Spektrum, Bd. 28, Nr. 4, 2005, S. 292-296.

Terry, D. et al. (1995): Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In: Proceedings of the 15th ACM Symposium on Operating Systems Principles, 1995, S. 172-183.

Tielert, R. (2004): Forschungsschwerpunkt Ambient Intelligence. Szenario: Assited Living, TU Kaiserslautern, http://www.eit.uni-kl.de/AmI/de/inhalte/assisted_living/_pics/Scenario_assisted_living.pdf, Abgerufen am 30.04.06, 2004.

Traas, V.; van Hillegersberg, J. (2000): The software component market on the internet current status and conditions for growth. In: SIGSOFT Software Engineering Notes, Bd. 25, Nr. 1, 2000, S. 114-117.

Tsichritzis, D. C.; Lug, A. (1978): The ANSI/X3/Sparc DBMS Framework - Report of the Study Group on Database Management Systems. In: Information systems, Bd. 3, Nr. 3, 1978, S. 173-191.

Turowski, K.; Conrad, S. (2001): Temporal OCL: Meeting Specifications Demands for Business Components. In: Siau, K.; Halpin, T. (Hrsg.): Unified Modeling Language: SystemsAnalysis, Design and Development Issues, Hershey, 2001, S. 151-165.

Turowski, K.; Pousttchi, K. (2004): Mobile Commerce: Grundlagen und Techniken, Berlin [u. a.], 2004.

Ullmer, B.; Ishii, H. (2000): Emerging Frameworks for Tangible unser Interfaces. In: IBM Systems Journal, Bd. 39, Nr. 3&4, 2000, 915-931.

Ulrich, P.; Fluri, E. (1995): Management: eine konzentrierte Einführung, 7. verb. Aufl., Bern [u. a.], 1995.

van Renesse, R.; Birman, K. P.; Vogels, W. (2003): Astrolabe: A robust and scalable Technology for distributed System Monitoring, Management, and Data Mining. In: CM Transactions on Computer Systems, Bd. 21, Nr. 2, 2003, S. 164-206.

Versteegen, G.; Salomon, K.; Heinold, R. (2001): Change-Management bei Softwareprojekten, Berlin [u. a.], 2001.

Versteegen, G. (2003): Konfigurationsmanagement, Berlin [u. a.], 2003.

Voas, J. (1998): COTS Software: The Economical Choice? In: IEEE Software, Bd. 15, Nr. 2, 1998, S. 16-19.

Vogler, M. (1994): OTMAR: Ein automatischer Übersetzer zur Konstruktion von Objekttypen und Beziehungen aus Aussagen, Diplomarbeit Universität Konstanz, Fakultät Informationsmanagement, Konstanz, 1994.

Vogt, W. (2002): Auch Mobilität hat ihre Grenzen. In: Der Bürger im Staat – Mobilität, Nr. 3, 2002.

Voigtmann, P.; Zeller, T. (2003): Beiträge zur Integrationsproblematik im Kontext von Electronic Business und Elektronischen Marktplätzen. In: Uhr, W.; Esswein, W.; Schoop, E. (Hrsg.): Wirtschaftsinformatik 2003 – Medien, Märkte, Mobilität, Bd. 1, Heidelberg [u. a.], 2003, S. 215-237.

W3C (1999): XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath, Abgerufen am 22.11.06, 1999.

W3C (2001): Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, Abgerufen am 22.11.06, 2001.

W3C (2004): Simple Object Access Protocol (SOAP) Specifications, http://www.w3.org/TR/soap/, Abgerufen am 22.11.06, 2004.

W3C (2005): HTTP - Hypertext Transfer Protocol, http://www.w3.org/Protocols/, Abgerufen am 08.12.06, 2005.

W3C (2006a): Extensible Markup Language (XML), http://www.w3.org/XML/, Abgerufen am 22.11.06, 2006.

W3C (2006b): HyperText Markup Language (HTML) Home Page, http://www.w3.org/MarkUp/, Abgerufen am 22.11.06, 2006.

W3C (2006c): World Wide Web Consortium, http://www.w3.org/, Abgerufen am 22.11.06, 2006.

W3C (2006d): XML Schema, http://www.w3.org/XML/Schema, Abgerufen am 22.11.06, 2006.

W3C (2006e): The Extensible Stylesheet Language Family (XSL), http://www.w3.org/Style/ XSL/, Abgerufen am 22.11.06, 2006.

WAP Forum (o. J.): WAP 2.0 Technical White Paper, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html, Abgerufen am 08.10.06, o. J.

Waldo, J. (1998): Javaspaces specification 1.0, http://www-csag.ucsd.edu/teaching/cse225s04/Reading%20List/microsystems98javaspaces.pdf, Abgerufen am 08.10.06, 1998.

Wang, N.; Schmidt, D. C.; O'Ryan, C. (2001): Overview of the CORBA Component Model. In: Heinemann, G. T.; Councill, W. T. (Hrsg.): Component-based Software Engineering, Upper Saddle River, 2001, S. 557-571.

Wedekind, H. et al. (1998): Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik. In: Informatik-Spektrum, Bd. 21, Nr. 5, 1998, S. 265-272.

Wedekind, H.; Ortner, E.; Inhetveen, R. (2004): Informatik als Grundbildung. In: Informatik Spektrum, Bd. 27, Nr. 4, 2004, S. 337-342.

Wedekind, H.; Ortner, E. (1980): Systematisches Konstruieren von Datenbankanwendungen: zur Methodologie der angewandten Informatik, München [u. a.], 1980.

Wedekind, H. (1973): Systemanalyse: die Entwicklung von Anwendungssystemen für Datenverarbeitungsanlagen, München, 1973.

Wedekind, H. (1999): Von der Informatik zum Computational Science and Engineering. In: Walther, A. (Hrsg.): Pionier des wissenschaftlichen Rechnens, TUD Schriftenreihe Wissenschaft und Technik, Bd. 75, Darmstadt, 1999.

Wedekind, H. (2005): Die Arbeitsteilung ist es! In: Informatik Spektrum, Bd. 28, Nr. 5, 2005, S. 440-442.

Weinreich, R.; Sametinger, J. (2001): Component Models and Component Services – Concepts and Principles. In: Heinemann, G. T.; Councill, W. T. (Hrsg.): Component-based Software Engineering, Upper Saddle River, 2001, S. 33-48.

Weiser, M. (1991): The Computer of the 21st Century. In: Scientific American, Bd. 91, Nr. 9, 1991, S. 94-100.

Weiser, M. (1993): Some Computer Science Issues in Ubiquitous Computing. In: Communications of the ACM, Bd. 36, Nr. 6, 1993, S. 75-84.

Westhoff, J. (2000): IDW-Informationsdienst Wissenschaft, Pressegespräch anlässlich des Kongresses Medizin und Mobilität http://idw-online.de/pages/de/ news?print=1&id=24482, Abgerufen am 19.03.06, 2000.

Westphal, R. (2002): .Net kompakt, Heidelberg [u. a.], 2002.

Wiecker, M. (2002): Breitbandige, kabellose Übertragungstechnologien. In: Gora, W.; Röttger-Gerigk, S. (Hrsg.): Handbuch Mobile-Commerce: technische Grundlagen, Marktchancen und Einsatzmöglichkeiten, Berlin [u. a.], 2002, S. 427-440.

Wiedenmaier, S. J. (2004): Unterstützung manueller Montage durch Augmented-reality-Technologien, Aachen, 2004.

Wild, J. (1971): Management-Konzeption und Unternehmungsverfassung. In: Schmidt, R.-B. (Hrsg.): Probleme der Unternehmungsverfassung, Tübingen, 1971, S. 57-95.

Wild, M.; Herges, S. (2000): Total Cost of Ownership (TCO) – Ein Überblick. In: Arbeitspapiere WI, Nr. 1, Mainz, 2000.

Wilkes, M. V. (1970): Time-sharing Computer Systems, Leipzig, 1970.

Wille, R.; Ganter, B. (1996): Formale Begriffsanalyse: mathematische Grundlagen, Berlin [u. a.], 1996.

Williams, J. D. (1996): Managing Iteration in OO Projects. In: Computer, Bd. 29, Nr. 9, 1996, S. 39-43.

Wilmsen, E. N. (1972): Social Exchange and Interaction, Ann Arbor, 1972.

Wollnik, M. (1988): Ein Referenzmodell des Informationsmanagements. In: Information Management, Bd. 3, Nr. 3, 1988, S. 34-43.

Wyckoff, P. et al. (1998): T spaces. In: IBM (Hrsg.): IBM Systems Journal, Bd. 37, Nr. 3, 1998, S. 454-474.

Zeidler, A.; Gruteser, M. (1999): Bezaubernde Geräte. In: iX – Magazin für professionelle Informationstechnik, Nr. 4, 1999, S. 144.

Zimmerman, J. B. (1999): Mobile Computing: Characteristics, Business Benefits, and the Mobile Framework, Bowie State, 1999.

Zimmermann, H.-D. et al. (2006): Electronic Markets – Special Issue Electronic Negotiation. In: Electronic Markets - The International Journal, Bd. 16, Nr. 3, 2006.

Zobel, J. (2001): Mobile-Business und M-Commerce: die Märkte der Zukunft erobern, München [u. a.], 2001.

Zwahr, A. (2000): Brockhaus, Sonderausgabe 2000, Leipzig [u. a.], 2000.

Zwicky, F. (1966): Entdecken, erfinden, forschen im morphologischen Weltbild, München [u. a.], 1966.

A Anhang – Systementwurf von mobiCOMP

Der grobe Aufbau des mobiCOMP-Systems wurde im Abschnitt 5.1 mit der Beschreibung der mobiCOMP-Architektur gezeigt. In Abbildung 58 wurden die einzelnen Marktplatzkomponenten veranschaulicht. Zwischen den Marktplatzkomponenten bestehen folgende Schnittstellen im System:

- Darstellungssystem (GUI) mit dem Vorgangssteuerungssystem
- Vorgangssteuerungssystem mit dem Kommunikationssystem
- Kommunikationssystem mit dem Dateitransfersystem
- Kommunikationssystem mit dem DB-Wrapper
- DB-Wrapper mit dem Metainformationssytem

Im Folgenden wird das methodenspezifische Systemkonzept wiedergegeben, das die Grundlage für die Implementierung von mobiCOMP bildet.

A.1 Das Darstellungssystem

A.1.1 Interne Struktur des Darstellungssystems

Das Darstellungssystem setzt sich im Wesentlichen aus Formularen, Programm-Skripten und die Schnittstelle zum VSS zusammen.

Aktionen des Benutzers werden an das VSS gesendet. Dieses verarbeitet die Aktion und sendet eine Formularidentifikationsnummer zurück. Daraufhin wird vom DSS das anzuzeigende Formular gerätespezifisch generiert. Zur Generierung des Formulars werden bestimmte Metadaten benötigt, welche die Struktur der anzuzeigenden Formularelemente beschreiben. Diese Daten werden über die VSS-Schnittstelle abgefragt und von einem zentralen Skript verarbeitet. Dieses Skript verteilt die Daten an formularspezifische Skripte, welche daraus den WML bzw. HTML Code für die Darstellung generieren. Nach Eingabe der Daten sendet der Benutzer das Formular ab. Vor der Weiterleitung der Eingaben an das VSS werden die Formulare auf syntaktischer Ebene auf Zulässigkeit überprüft. Dazu gehört die Überprüfung, ob

- alle Muss-Felder ausgefüllt wurden,
- Textfelder und Textbereiche die vorgeschriebene Länge einhalten,
- Datumsangaben im vorgesehenen Bereich liegen,
- E-Mail Adressen eine gültige Struktur haben und
- ob Zahlen im vorgeschrieben Bereich liegen.

Sollte mindestens eine der Eingaben ungültig sein, wird das Formular mit einem entsprechenden Hinweis erneut angezeigt. Sobald alle Eingaben korrekt vorliegen, werden diese an das VSS weitergeleitet.

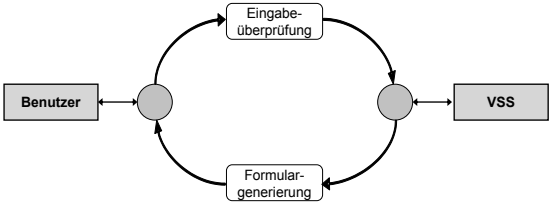

Abbildung 60: Interaktion[622] Benutzer – Vorgangssteuerungssystem

Formulardefinitionen

Formular	**Willkommenseite**	
Beschreibung	Willkommenseite für den mobiCOMP Marktplatz	
Felder	*keine*	

Formular	**Authentifikation**	
Beschreibung	Formular zur Eingabe der Authentifikationsdaten des Benutzers	
Felder	Benutzername	Textfeld
	Passwort	Textfeld

Formular	**Komponente suchen**	
Beschreibung	Ermöglicht die Suche nach Komponenten basierend auf verschiedenen Kriterien	
Felder	Komponentenname	Textfeld
	Preis	Textfeld
	Preis-Bedingung	Liste
	Währung	Liste
	Betriebssystem	Liste
	Hersteller	Textfeld

[622] Zur Notation von Interaktionsdiagrammen vgl. Oestereich 2005, S. 326.

Formular	**Komponente stöbern**	
Beschreibung	Zeigt die für die Komponenten vorgesehenen Kategorien zum Stöbern nach Komponenten an	
Felder	*Verweise für Kategorien*	

Formular	**Komponentendetails anzeigen**	
Beschreibung	Zeigt erweiterte Details zu einer Komponente an	
Felder	Komponentenname	Textfeld
	Preis	Textfeld
	Währung	Textfeld
	Beschreibung	Textfeld

Formular	**Komponente kaufen**	
Beschreibung	Fragt den Benutzer nach einer Bestätigung vor dem Kaufen einer Komponente	
Felder	*Bestätigungsseite*	

Formular	**Komponente hinzufügen**	
Beschreibung	Erlaubt dem Benutzer das Hinzufügen einer Komponente auf dem Marktplatz	
Felder	Komponentenname	Textfeld
	Beschreibung	Textfeld
	Preis	Textfeld
	Betriebssystem	Liste
	Währung	Liste

Formular	**Komponente ändern**	
Beschreibung	Erlaubt einem Benutzer die Daten zu einer Komponente zu ändern	
Felder	*siehe Komponente hinzufügen*	

Formular	**Registrierung**	
Beschreibung	Formular zur Eingabe der Registrierungsseite eines neuen Benutzers und zum Ändern der Registrierungsdaten eines vorhandenen Benutzers	
Felder	Vorname	Textfeld
	Nachname	Textfeld
	Telefonnummer	Textfeld
	Faxnummer	Textfeld
	Postleitzahl	Textfeld
	Wohnort	Textfeld
	Kreditkartennummer	Textfeld
	Kreditkartengültigkeit	Textfeld
	Kreditkartengesellschaft	Liste
	E-Mail-Adresse	Textfeld

Formular	**Statistik anzeigen**	
Beschreibung	Zeigt detaillierte Statistiken zu einer Komponente an	
Felder	Komponentenname	Textfeld
	Anzahl Zugriffe auf Detailseite	Textfeld
	Anzahl Käufe	Textfeld
	Anzahl Downloads	Textfeld
	Erstellungsdatum	Textfeld
	Letzte Änderung	Textfeld
	Anzahl Änderungen insgesamt	Textfeld

Formular	**Verabschiedung**	
Beschreibung	Verabschiedungsseite für den mobiCOMP Marktplatz	
Felder	*keine*	

Metadaten

Im Folgenden werden die Datentypen für die Metadaten spezifiziert.

FormData	
FormObject] formObject	Die Formularfelder werden durch das 2D-Array gruppiert, wobei die erste Dimension des Arrays die Gruppe und die zweite Dimension Elemente in dieser Gruppe bezeichnet.

FormObject	
String title	Titel des Feldes
String data	aktuelle Daten
Boolean mandatory	Pflichtfeld?

TextField::FormObject	
TextFieldType type	Definiert Typ des Textfeldes

TFTNumeric::TextFieldType	
int min	kleinste gültige Zahl
int max	größte gültige Zahl

TFTAlphanumeric::TextFieldType	
int maxLength	maximale Anzahl an akzeptierten Zeichen

TFTEmail::TextFieldType

TFTDate::TextFieldType	
date from	kleinstes akzeptiertes Datum
date to	größtes akzeptiertes Datum

TextArea::FormObject	
int maxLength	Anzahl akzeptierter Zeichen

List::FormObject	
Boolean multiple	Mehrfachauswahl möglich
ListEntry listEntry	Liste mit Einträgen

ListEntry	
int ID	Identifikation für Listenauswahl
String label	Darstellung (Text) der Listenauswahl

Klassendiagramme der Metadaten

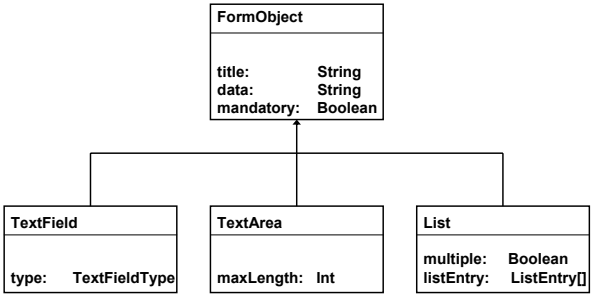

Abbildung 61: Klassendiagramm[623] Formular Objekt

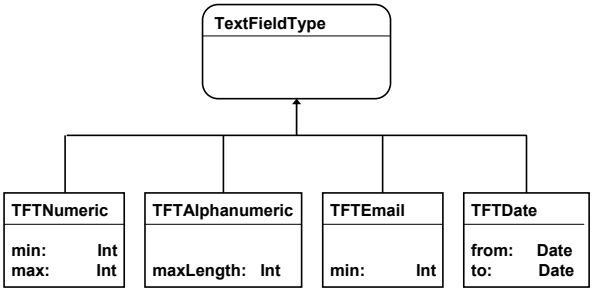

Abbildung 62: Klassendiagramm Textfeldtyp

[623] Zur Notation von Klassendiagrammen vgl. Hitz et al. 2005, S. 52ff.

A.2 Die Objekte des mobiCOMP Marktplatzsystems

User Data

Attribute	Typ	Beschreibung
Company	String	Firmenname
Prename	String	Vorname
Name	String	Name
Phone	Int	Telefonnummer
Fax	Int	Optional; Faxnummer
Plz	Int	Postleitzahl
City	String	Stadt
creditCardNumber	Int	Optional; Kreditkartennummer
bankName	String	Optional; Bankname
Blz	Int	Optional; Bankleitzahl
Account	Int	Optional; Kontonummer
Payment	Choice	Auswahl der Zahlungsart
Password	String	Passwort
sessionID	Session	Siehe Objekt Session
actionID	String	Kann folgende ActionID haben: • Registration • ChangeUserData

Authentification

Attribute	Typ	Beschreibung
userID	String	Siehe Objekt Session
Password	String	Passwort
actionID	String	Kann folgende ActionID haben: • Login • RemoveUser

Session

Attribute	Typ	Beschreibung
sessionID	String	Die eindeutige Identifikation der Session
actionID	String	Kann folgende ActionID haben: • CreateSession • UpdateSession
UserID	String	Die eindeutige Identifikation des Benutzers

WebPage

Attribute	Typ	Beschreibung
webPageID	String	Eine eindeutige Identifikation einer Webseite
actionID	String	Kann folgende ActionID haben: • ShowWebPage

Component

Attribute	Typ	Beschreibung
componentName	String	Dem Benutzer angezeigter Komponentenname
Price	Double	Preis der Komponente
OS	String	Benötigtes Betriebssystem
Description	String	Beschreibung der Komponente
smallDescription	String	Kurze Beschreibung
requiredFeatures	String	Funktionale Voraussetzungen
Version	String	Version
Currency	Int	Währung
producerName	String	Herstellername der Komponente
componentID	String	Eindeutige Identifikation der Komponente
sessionID	Session	Siehe Objekt Session
Component	Binary	Optional; Die Komponente als Binary
actionID	String	Kann folgende ActionID haben: • AddComponent • ChangeComponent

DownloadComponent

Attribute	Typ	Beschreibung
sessionID	Session	Siehe Objekt Session
componentID	String	Eindeutige Identifikation der Komponente
actionID	String	Kann folgende ActionID haben: • DownloadComponent
Component	Binary	Die Komponente als Binary

Category

Attribute	Typ	Beschreibung
categoryID	String	Die eindeutige Identifikationsnummer der Kategorie
categoryName	String	Dem Benutzer angezeigter Kategoriename
Description	String	Beschreibung dieser Kategorie; vom Benutzer zu lesen
subCategories	Array SubCatego-ry	Die Beschreibung der Unterkategorien
components	Array Component	Optional
actionID	String	Kann folgende ActionID haben: • BrowseComponent
sessionID	Session	Siehe Objekt Session

SubCategory

Attribute	Typ	Beschreibung
categoryName	String	Der dem Benutzer angezeigte Name einer Kategorie
categoryID	String	Die eindeutige Identifikationsnummer der Kategorie

SearchComponent

Attribute	Typ	Beschreibung
componentName	String	Nach diesen Namensfragmenten soll gesucht werden
Price	Double	Preis der Komponente; steht in Verbindung mit priceLimit
OS	String	Gewünschtes Betriebssystem der Komponente
description	String	Suche in der Beschreibung der Komponente
priceLimit	String	Wie soll der Preis sein: less, greater, equal
producerName	String	Sucher nach Herstellernamen
componentID	String	Wenn man die ID kennt wird nach ihr gesucht
sessionID	Session	Siehe Session
actionID	String	Kann folgende ActionID haben: • SearchComponent

A.3 Schnittstelle Darstellungssystem ↔ Vorgangssteuerungssystem

Für die Schnittstelle DSS/VSS werden die Methoden performAction und getForm bereitgestellt.

A.3.1 Methode performAction

Spezifikation	performAction (ActionData data, ActionID actionID, UserID userID)
Parameter	*data*: Daten, welche die Aktion betreffen *actionID*: Eindeutiger Bezeichner für duchzuführende Aktion *userID*: ID des Benutzers, für den die Aktion durchgeführt werden soll
Rückgabe	*FormID*, zu dem anschließend verzweigt wird; hierdurch kann auch zu einer Seite zur Anzeige von Fehlermeldungen verzweigt werden

Über die Methode performAction leitet die GUI die Aktionen des Anwenders an das VSS weiter. Durch den mitgelieferten Parameter ActionData können optional an einen Aufruf Eingabedaten angehängt werden. Bei den Eingabedaten handelt es sich um Daten, die der Anwender dem System über die grafische Oberfläche (GUI) zur Weiterverarbeitung zur Verfügung stellt. Diese Daten werden an die nachfolgende Ebene weitergegeben.

Neben der ActionData erhält das VSS über die Parameter ActionID und UserID weitere Informationen. Während die UserID dazu dient, die Aktionen des Systems eindeutig dem Anwender zuzuordnen, ermöglicht die ActionID die Einordnung der ausgeführten Aktion. Diese Einordnung erfolgt über eine Konvention.

Anhand der mitgelieferten Parameter ist es dem VSS nun möglich den Anwender zu identifizieren, seinen aktuellen Aufenthalt auf dem Marktplatz (i. S. von Dialog) zu

bestimmen und seine ausgeführten Aktionen zu registieren. Aus diesen drei Parametern kann der nächste Schritt im zugrundeliegenden Vorgang eingeleitet werden.

Die FormID ist einer inhaltlosen Seite zugeordnet, deren Inhalt über die Methode getForm ermittelt werden soll (siehe Methodenbeschreibung getForm in Abschnitt 0).

Parameterspezifikation

Im Folgenden werden mögliche Aktionen, die im Zusammenhang mit der Methode performAction eine Rolle spielen, näher beschrieben. Die Parameterspezifikation ist abhängig von ActionID.

ActionID	**Login**	**Typ**
ActionData	userName	String
	password	String
UserID	sessionID	String
Beschreibung	Benutzer einloggen	

ActionID	**Logout**	**Typ**
ActionData	Null	
UserID	sessionID	String
Beschreibung	Benutzer ausloggen	

ActionID	**RemoveUser**	**Typ**
ActionData	Null	
UserID	sessionID	String
Beschreibung	Benutzer entfernen	

ActionID	**Registration**	**Typ**
ActionData	prename	String
	lastname	String
	phone	String
	fax	String
	postCode	String
	place	String
	creditcardNr	String
UserID	sessionID	String
Beschreibung	Registrierung eines Benutzers im System	

ActionID	**AddComponent**	**Typ**
ActionData	componentName	String
	requiredOS	Int
	… (Daten entsprechend DB)	
UserID	sessionID	String
Beschreibung	Eine neue Komponente hinzufügen	

ActionID	**SearchComponent**	**Typ**
ActionData	componentName	String
	price	Double
	priceCond	{less, greater, equal}
	currency	Int
	requiredOS	Int
	producerName	String
UserID	sessionID	String
Beschreibung	Stichwortsuche nach mehreren Kriterien	

ActionID	**ChangeComponent**	**Typ**
ActionData	componentName	String
	requiredOS	Int
	… (Daten entsprechend DB)	
UserID	sessionID	String
Beschreibung	Komponentendaten aktualisieren	

ActionID	**BrowseComponent**	**Typ**
ActionData	category	Int
	accessPath	Int
	(Pfad auf dem „gestöbert“ wird, d. h. spezifiziert auf welchem Weg Benutzer gerade stöbert) Bspw.: AccessPath=(2, 3, 5) ? auf 1. Ebene, 2. Eintrag gewählt auf 2. Ebene, 3. Eintrag gewählt auf 3. Ebene, 5. Eintrag gewählt Nutzer sieht alle Einträge von Ebene 4	
UserID	sessionID	String
Beschreibung	Ebenensuche mit Katalogansicht	

ActionID	**ShowComponent**	**Typ**
ActionData	componentID	Int
UserID	sessionID	String
Beschreibung	Details zu einer Komponente anzeigen	

ActionID	**ShowStatistics**	**Typ**
ActionData	componentID	Int
	Formulardaten für Statistik beinhalten:	
	views	Int
	(Anzahl Zugriffe auf Detail Ansicht der Komponente)	
	buys	Int (Anzahl Käufe)
	downloads	Int
	creationDate	Date
	lastChange	Date
	changesTotal	Int
	(Anzahl Änderungen an Komponente)	
UserID	sessionID	String
Beschreibung	Verkaufsstatistik anzeigen	

ActionID	BuyComponent	Typ
ActionData	componentID	Int
UserID	sessionID	String
Beschreibung	Komponente kaufen	

ActionID	DownloadComponent	Typ
ActionData	componentID	Int
UserID	sessionID	String
Beschreibung	Komponente herunterladen	

A.3.2 Methode getForm

Spezifikation	getForm (FormID formID)
Parameter	FormID: Identifikator für Formular
Rückgabe	FormData

Die Methode getForm liefert Metadaten und evtl. auch den Inhalt zur Darstellung eines Formulars. Handelt es sich um eine Datenneueingabe durch den Benutzer, wie bspw. bei der Registrierung, so liefert getForm lediglich die Metadaten. Bei einer Datenänderung hingegen, wie bspw. bei einer Registrierungsdatenänderung, liefert die Methode sowohl Metadaten als auch deren Inhalt. An die Metadaten und deren Inhalt gelangt getForm über eine Schnittstelle zum KDS (Kommunikations- und Datentransfersystem).

A.4 Schnittstelle Vorgangssteuerungssystem ↔ Kommunikationssystem

Für die Schnittstelle VSS/KSS wird die Methoden performAction bereitgestellt.

A.4.1 Methode performAction

Attribut	Ausprägung
Methodenname	performAction
Parameter	Objekt: Die Methode beinhaltet nur einen Parameter, der von verschiedenen Typen sein kann. Jedes übergebene Objekt muss die Attribute SessionID und ActionID beinhalten. Anhand dieser Attribute wird der aufzurufende Workflow erkannt.
Rückgabe	Objekt: Die Rückgabe ist ein Objekt der gleichen Struktur oder eine Fehlermeldung.

Das Vorgangssteuerungssystem leitet Anfragen und Aufgaben über die Methode performAction an das Kommunikationssystem weiter. Das Kommunikationssystem gliedert diese Aufgaben und verteilt die Teilaufgaben an weitere Systeme. Es wird nur eine Methode als Schnittstelle angeboten, anhand der ActionID kann festgestellt werden welcher Workflow gerade angestoßen wird. In den übergebenen Objekten sind

Metadaten enthalten. Die Metadaten beschreiben wie die Daten im Webinterface angezeigt werden sollen.

ActionID	Beschreibung
Login	Einloggen eines Benutzers. UserID und Passwort werden gebraucht.
Logout	Der Benutzer hat sich ausgeloggt.
Registration	Der Registrations Workflow wurde angestoßen.
SearchComponent	Eine Suche nach Komponenten soll durchgeführt werden. Es handelt sich um eine Suche nach bestimmten Parametern.
BrowseComponent	Es wird im Katalog nach Komponenten gesucht.
AddComponent	Eine Komponente soll hinzugefügt werden.
ChangeComponent	Eine Komponente soll geändert werden.
BuyComponent	Die ausgewählte Komponente soll gekauft werden.
RemoveUser	Der Benutzer möchte seine Registrierung vom Marktplatz löschen.
DownloadComponent	Ein Download von Komponenten wird angestoßen.
ChangeUserData	Der Benutzer möchte seine Daten ändern.
CreateSession	Wenn sich ein Benutzer am Marktplatz anmeldet, muss eine neue Session erstellt werden.
UpdateSession	Die aktuelle Session muss erneuert werden; z. B. weil sich der Benutzername geändert hat.
ShowStatistics	Die Statistiken sollen angezeigt werden.

Objekte

Die mit den ActionIDs im Zusammenhang stehenden Objekte (siehe Abschnitt A.2) sind: User Data, Authentification, Component, DownloadComponent, Category, Session, SubCategory, SearchComponent und WebPage

A.5 Schnittstelle Kommunikationssystem ↔ Dateitransfersystem

Für die Schnittstelle KSS/DTS wird die Methode performAction bereitgestellt.

A.5.1 Methode performAction

Spezifikation	Methode performAction
Parameter	DownloadComponent object
Rückgabe	Fehlermeldung oder DownloadComponent Objekt

Über die Methode performAction werden sämtliche Funktionen des Dateitransfersystems aufgerufen. Die übergebenen Objekte haben sessionID und actionID als Attribute gemeinsam. Anhand dieser Attribute kann festgestellt werden welche Aktion ausgeführt werden soll.

ActionID	Beschreibung
show	Ein Download soll gestartet werden.
delete	Ein Download ist erfolgreich abgeschlossen.
abort	Ein Download wurde abgebrochen.

Objekte

Das mit den ActionIDs im Zusammenhang stehende Objekt (siehe Abschnitt A.2) ist: DownloadComponent.

A.6 Schnittstelle Kommunikationssystem ↔ Datenbank-Wrapper

Für die Schnittstelle KSS/DBWrapper wird die Methode action bereitgestellt:

A.6.1 Methode action

Diese Methode ist die einzige Methode, die der DB-Wrapper bereitstellt. Verschiedene Funktionalitäten werden über unterschiedliche Paramter und ActionIDs initialisiert. Dementsprechend ist diese Methode überladen. Das Rückgabeobjekt der Methode richtet sich nach der ActionID.

Spezifikation	Methode action
Parameter	Object object
Rückgabe	Object object

Die Parameter können vom Typ

a. Component
b. Category
c. Authentification
d. DownloadComponent
e. UserData
f. WebPage
g. FormularLayout
h. Session

sein.

Rückgabe: Jedes Objekt enthält eine Variable actionID, je nach Wert der Variable kann das Rückgabeobjekt vom folgenden Typ sein:

a. Component
b. Category

c. Authentification
d. DownloadComponent
e. UserData
f. WebPage
g. FormularLayout
h. Void

Das konkrete Rückgabeobjekt kann aus folgender Tabelle entnommen werden.

ActionID	Beschreibung
change	Es soll ein Datensatz geändert werden.
insert	Es soll ein neuer Datensatz eingefügt werden.
delete	Es soll ein Datensatz gelöscht werden.
abort	Es soll ein Vorgang abgebrochen werden.
show	Es sollen Daten angezeigt werden.

Objekte

Die mit den ActionIDs im Zusammenhang stehenden Objekte (siehe Abschnitt A.2) sind: User Data, Authentification, Component, DownloadComponent, Category, Session, SubCategory, SearchComponent und WebPage.

Im Folgenden sollen mögliche Aktionen, die im Zusammenhang mit der Methode action eine Rolle spielen, näher beschrieben werden. Die Parameterspezifikation ist abhängig von ActionID.

Alle ActionIDs sind Strings

Objects	Action Ids	Rückgabeobjekt	Kommentar
Component	insert	Void	
	show	Component	Suche nach Komponenten; je nach gefüllten Attributen
	delete	Void	
	change	Void	
UserData	insert	Void	Registrierung
	show	UserData	
	delete	Void	
	change	Void	
DownloadComponent	show	DownloadComponent	show=download durchführen
	delete	Void	nach erfolgreichem Download
	abort	Void	
Category	show	Category	
Authentification	show	Authentification	Übergeben der User Ids

A.7 Dokumentation der Schemata und Metaschemata

A.7.1 DBKat – Komponentenkatalog-Datenbank

Die Komponentenbeschreibung erfolgt mit Hilfe des Spezifikationsrahmens der GI[624] für Fachkomponenten (siehe Abschnitt 3.5.2.5). Daher umfasst DBKat alle notwendigen Relationen zur Umsetzung des Spezifikationsrahmens. Diese Spezifikation basiert auf sieben Beschreibungsebenen, die im Folgenden einzeln behandelt werden.

A.7.1.1 Vermarktungsebene

Die Vermarktungsebene beinhaltet im wesentlichen vier Kategorien von Daten: grundlegende Informationen über abzulegende Komponenten wie Komponentenname und –version, Informationen über Bezugsmodalitäten (bspw. Komponentenpreis, Kontaktdaten und Vertragsbedingungen), klassifizierende Daten, welche die hierarchische Kategorisierung nach Wirtschaftszweig und Anwendungsdomäne ermöglichen sowie Anforderungen an die zugrunde liegende Systemarchitektur (untergliederbar in Hardware, Software und Basissysteme).

Relationenschemata

„Component_Type“	***Art der Komponente (Fach-/Systemkomponente, Framework)***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Type	Serial	☑	☐		
2	FatherType	Int	☐	☑	Component_Type.ID_Type	*optional*
3	Name	VarChar	☐	☐		
4	Description	CLOB	☐	☐		*optional*
Integritätsbedingungen: ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherType“ existieren*						

Anmerkungen:

- verwendete Fachbegriffe: ComponentType, Fachkomponente, Systemkomponente, Framework, Komponenten-Anwendungs-Framework, Komponenten-System-Framework
- „FatherType“ ermöglicht eine hierarchische Strukturierung der Komponententypen.

[624] Vgl. Ackermann et al. 2002.

„Component“		***Hauptdatenstruktur einer abzulegenden Software-Komponente***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Component	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Version	VarChar	☐	☐		*optional*
4	Price	Dec(10,2)	☐	☐		*Price ≥ 0*
5	Currency	Int	☐	☑	Component_Currency.ID_Currency	*optional*
6	Company	Int	☐	☑	ContactPersonFor.Company, Company.ID_Company	*optional*
7	Person	Int	☐	☑	ContactPersonFor.Person, Person.ID_Person	*optional*
8	BankAccount	Int	☐	☑	BankAccount.ID_BankAccount	*optional*
9	Description	CLOB	☐	☐		*optional*
10	ContractualConditions	CLOB	☐	☐		*optional*
11	Size	Int	☐	☐		*Size ≥ 0*
12	ComponentType	Int	☐	☑	Component_Type.ID_Type	
13	UsedTechnology	Int	☐	☑	Component_Technology.ID_Technology	*optional*
14	UsedFramework	Int	☐	☑	Component_Framework.ID_Framework	*optional*
15	UsedDBMS	Int	☐	☑	Component_DBMS.ID_DBMS	*optional*
16	RequiredCPU	VarChar	☐	☐		*optional*
17	RequiredMM	Int	☐	☐		*optional, MM ≥ 0*
18	RequiredHDD	Int	☐	☐		*HDD ≥ 0*
19	RequiredOS	Int	☐	☑	Component_OS.ID_OS	
Integritätsbedingungen: *¬(Price = 0) ∨ ((Currency ist gültiger Verweis) ∧ (BankAccount ist gültiger Verweis))*						

Anmerkungen:

- verwendete Fachbegriffe: Component, ProducingCompany, ContactPerson, ComponentType, Technology, Framework, DBMS, OS
- verwendete Abkürzungen: DBMS, CPU, MM, HDD, OS
- „UsedFramework“ ist optional, da Frameworks auch Komponenten darstellen, selbst aber in der Regel auf kein anderes Framework zurückgreifen.

„Component_Storage“	***Physische Speicherung der Komponente***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Component	Int	☑	☑	Component.ID_Component	
2	Data	BLOB	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component
- Um flexible Erweiterbarkeit und Gestaltungsmöglichkeiten zu eröffnen, werden die physischen Komponentendaten getrennt von den Beschreibungsinformationen gehalten.

„Component_Currency“	***Währungen, in denen Komponentenpreise angegeben sind***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Currency	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Country	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen: (keine)

„Component_Technology“	***Technologien, auf denen die Komponenten basieren***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Technology	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Producer	VarChar	☐	☐		
4	Version	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Technology
- „Producer“ korrelieren nicht mit „Companies“ aus DBUser; sie werden in DBKat nicht weiter geführt.

„Component_Framework“	***Frameworks, auf denen die Komponenten basieren***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Framework	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Producer	VarChar	☐	☐		
4	Version	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Framework, Komponenten-Anwendungs-Framework, Komponenten-System-Framework
- „Producer“ korrelieren nicht mit „Companies“ aus DBUser; sie werden in DBKat nicht weiter geführt.

„Component_ScopeOfSupply“	***Lieferumfang der Komponenten***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_SupplyNumber	Serial	☑	☐		
2	Component	Int	☐	☑	Component.ID_Component	
3	Description	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component
- Beispiele für Lieferumfang (neben der eigentlichen Komponente) sind Handbücher, Workshop-Gutscheine, Beratungsdienstleistungen, Einführungsvideos etc.

„Component_DBMS“	***Datenbank-Systeme, die von den Komponenten benötigt werden***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_DBMS	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Producer	VarChar	☐	☐		
4	Version	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: DBMS
- verwendete Abkürzungen: DBMS
- „Producer“ korrelieren nicht mit „Companies“ aus DBUser; sie werden in DBKat nicht weiter geführt.

„Compo-nent_Domain“	***Anwendungsdomänen der Komponenten (m:n - Materiali-sierung)***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	Domain	Int	☑	☑	Component_DomainStructure.ID_Domain	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Domain
- Eine Domäne kann viele zugeordnete Komponenten besitzen; eine Komponente kann jedoch auch mehreren Domänen zugeordnet sein.

„Component_OS“		***Betriebssysteme, die von den Komponenten benötigt werden***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_OS	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Producer	VarChar	☐	☐		
4	Version	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: OS
- verwendete Abkürzungen: OS
- „Producer“ korrelieren nicht mit „Companies“ aus DBUser; sie werden in DBKat nicht weiter geführt.

„Component_DomainStructure“		***Anwendungsdomänen der Komponenten als Baumstruktur***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Domain	Serial	☑	☐		
2	FatherDomain	Int	☐	☑	Component_DomainStructure.ID_Domain	*optional*
3	Name	VarChar	☐	☐		
4	Description	CLOB	☐	☐		
Integritätsbedingungen: ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherDomain“ existieren*						

Anmerkungen:

- verwendete Fachbegriffe: Domain
- „FatherDomain“ ermöglicht eine hierarchische Strukturierung der Domänen.

„Component_IndustrialSector“			*Wirtschaftszweige der Komponenten (m:n - Materialisierung)*			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	IndustrialSector	Int	☑	☑	Component_IndustrialSector. ID_IndustrialSector	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, IndustrialSector
- Ein Wirtschaftszweig kann viele zugeordnete Komponenten besitzen; eine Komponente kann jedoch auch mehreren Wirtschaftszweigen zugeordnet sein.

„Component_IndustrialSectorStructure“			*Wirtschaftszweige der Komponenten als Baumstruktur*			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_IndustrialSector	Serial	☑	☐		
2	FatherIndustrialSector	Int	☐	☑	Component_IndustrialSector. ID_IndustrialSector	*optional*
3	Name	VarChar	☐	☐		
4	Description	CLOB	☐	☐		
Integritätsbedingungen: ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherIndustrialSector“ existieren*						

Anmerkungen:

- verwendete Fachbegriffe: IndustrialSector
- „FatherIndustrialSector“ ermöglicht eine hierarchische Strukturierung der Wirtschaftszweige.

Es existieren keine interrelationalen Integritätsbedingungen in der Vermarktungsebene.

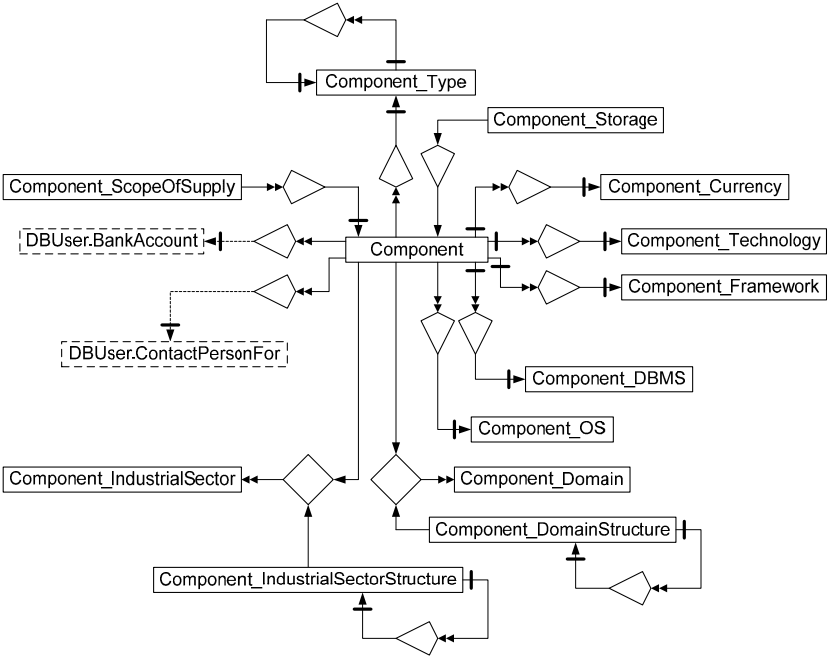

Abbildung 63: Objekttypendiagramm[625] DBKat – Vermarktungsebene

A.7.1.2 Aufgabenebene

Die Aufgabenebene erfasst normsprachlich die Funktionalität der Komponente, wobei von den einzelnen, an der Schnittstelle bereitgestellten Prozeduren, abstrahiert wird. Aufgaben können hierbei hierarchisch (sowohl abstraktiv als auch kompositiv) beliebig zerlegt und von entsprechenden Prozeduren der Schnittstelle referenziert werden. Hierbei wird von DBNorm Gebrauch gemacht, welche die Normsprachenfunktionalität zur Verfügung stellt.

[625] Zur Notation vgl. Mertens 2001, S. 342f., Stichwort „Objekttypenmethode“.

Relationenschemata

„Component_Tasks“		***Normsprachlich beschriebene Aufgaben der Komponenten***			
#	**Attribut**		**Schlüsseleigenschaften**		**Integritätsbedingungen**
	Name	Typ	PK	FK	
				Verweis	
1	Component	Int	☑	☑	Component.ID_Component
2	ID_SentenceOfComponent	Int	☑	☐	
3	SentenceStructure	Int	☐	☑	SentenceStructure. ID_SentenceStructure
4	FatherTask	Int	☐	☑	Component_Tasks. ID_SentenceOfComponent
Integritätsbedingungen: ➢ *ID_SentenceOfComponent muss innerhalb jeder Komponente eindeutig sein* ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherTask“ existieren*					

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Tasks, SentenceStructure
- „FatherTask“ ermöglicht eine hierarchische Strukturierung der Aufgaben.

„Component_Task_Terms“			***Materialisierte m:n - Zuordnung von Fachbegriffen zu den zur Aufgabenbeschreibung der Komponenten verwendeten Satzbauplänen***			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component_Tasks.Component	
2	SentenceOfComponent	Int	☑	☑	Component_Tasks.ID_SentenceOfComponent	
3	PlaceholderPosition	Int	☑	☐		
4	PlaceholderTerm	Int	☐	☑	Term.ID_Term	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Tasks, SentenceStructure, Placeholder, Term

Interrelationale Integritätsbedingungen

- „PlaceholderPosition“ in „Task_Terms“ muss einen Wert zwischen „1“ und „NoOfPlaceholders“ des entsprechenden Eintrags in „SentenceStructure“ besitzen:
- 1 ≤ Task_Terms[comp,sent].PlaceholderPosition ≤ SentenceStructure[Task_Terms [comp,sent].SentenceStructure].NoOfPlaceholders

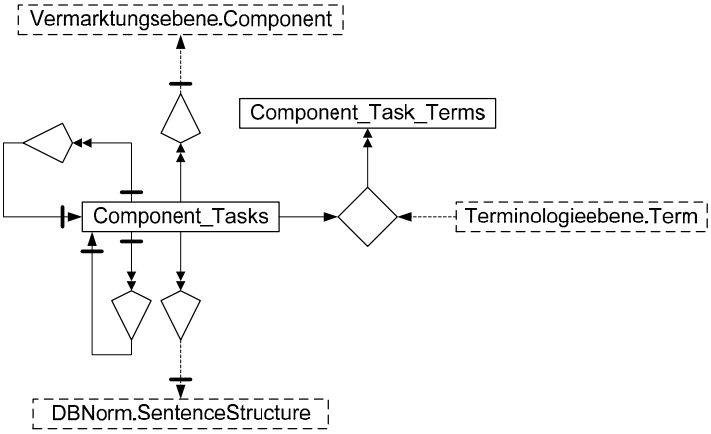

Abbildung 64: Objekttypendiagramm DBKat – Aufgabenebene

A.7.1.3 Terminologieebene

Die Terminologieebene des GI-Spezifikationsrahmens ist Teil der Normsprache und daher in DBNorm enthalten.

A.7.1.4 Qualitätsebene

Die Qualitätsebene erlaubt die hierarchische Speicherung von Qualitätsklassen mit zugeordneten Qualitätseigenschaften, welche wahlweise für eine gesamte Komponente oder für einzelne Komponentenfunktionen gelten kann. In beiden Fällen werden Qualitätsausprägungen gespeichert, welche sich zwecks Vergleichbarkeit explizit auf im System vorgegebene Referenzumgebungen beziehen.

Relationenschemata

„QualityClass“		***Qualitätsklassen der Komponenten***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_QualityClass	Serial	☑	☐		
2	Father	Int	☐	☑	QualityClass.ID_QualityClass	
3	Name	VarChar	☐	☐		
4	ShortDescription	VarChar	☐	☐		
5	FullDescription	CLOB	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: QualityClass
- „Father“ ermöglicht eine hierarchische Strukturierung der Qualitätsklassen.

„QualityProperty“		***Qualitätsmerkmale der Qualitätsklassen***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_QualityProperty	Serial	☑	☐		
2	QualityClass	Int	☐	☑	QualityClass.ID_QualityClass	
3	DynamicProperty	Boolean	☐	☐		
4	PropertyForComponent	Boolean	☐	☐		
5	ShortDescription	VarChar	☐	☐		
6	FullDescription	CLOB	☐	☐		
7	UnitOfMeasurement	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: QualityClass, QualityProperty, DynamicProperty

„QualityOfComponent“			***Ausprägung eines komponentenbezogenen Qualitätsmerkmals als materialisierte ternäre Beziehung***			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component_Interface_Function.Component	
2	QualityProperty	Int	☑	☑	QualityProperty.ID_QualityProperty	
3	TestEnvironment	Int	☑	☑	TestEnvironment.ID_TestEnvironment	
4	Value	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, QualityProperty, TestEnvironment

„QualityOfFunction“			***Ausprägung eines funktionsbezogenen Qualitätsmerkmals als materialisierte ternäre Beziehung***			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component_Interface_Function.Component	
2	NameOfFunction	VarChar	☑	☑	Component_Interface_Function.Name	
3	QualityProperty	Int	☑	☑	QualityProperty.ID_QualityProperty	
4	TestEnvironment	Int	☑	☑	TestEnvironment.ID_TestEnvironment	
5	Value	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface_Function, QualityProperty, TestEnvironment

„TestEnvironment“		*Testumgebung für Qualitätsmerkmale*				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_TestEnvironment	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Description	CLOB	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: TestEnvironment

Interrelationale Integritätsbedingungen

- $\text{QualityProperty[i].PropertyForComponent} \rightarrow \neg\exists \text{j.QualityOfFunction[j].QualityProperty} = \text{i}$
- $\neg\text{QualityPr operty[i].PropertyForComponent} \rightarrow \neg\exists \text{j.QualityOfComponent[j].QualityProperty} = \text{i}$

Objekttypendiagramm siehe Abbildung 65.

A.7.1.5 Abstimmungsebene

Die Abstimmungsebene hat zum Ziel, die Reihenfolgebeziehungen zwischen den Funktionen der Komponenten in formaler Notation (TemporalOCL, siehe Abschnitt 3.5.2.9) zu spezifizieren.

Relationenschemata

„Component_Coordination“		*Reihenfolgebeziehungen zw. den Diensten der Komponenten*				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	TemporalOCL	CLOB	☐	☐		*Stellt eine wohlgeformte, semantisch korrekte TemporalOCL-Datei dar*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Coordination, TemporalOCL

Es existieren keine interrelationalen Integritätsbedingungen in der Abstimmungsebene.

Objekttypendiagram siehe Abbildung 65.

A.7.1.6 Verhaltensebene

Die Verhaltensebene spezifiziert in formaler Notation (OCL, siehe Abschnitt 3.5.2.8) die Vor- und Nachbedingungen sowie Invarianten der Komponenten.

Relationenschemata

„Component_Behaviour“	***Vor- und Nachbedingungen sowie Invarianten der Komponenten***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	OCL	CLOB	☐	☐		*Stellt eine wohlgeformte, semantisch korrekte OCL-Datei dar*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Behaviour, OCL

Es existieren keine interrelationalen Integritätsbedingungen in der Verhaltensebene.

Objekttypendiagramm siehe Abbildung 65.

A.7.1.7 Schnittstellenebene

Die Schnittstellenebene umfasst neben der direkt von Programmierumgebungen verwendbaren IDL der Komponenten die Zuordnung von IDL-Funktionen zu Aufgaben der Aufgabenebene sowie von IDL-Datenobjekten zu Termini der Terminologieebene, wodurch semantische Korrektheit gewährleistet werden kann. Eine IDL-Funktion kann hierbei mehreren Aufgaben zugeordnet werden, ein Datenobjekt jedoch nur mit einem Terminus übereinstimmen.

Relationenschemata

„Component_Interface“	***Beschreibung der Komponentenschnittstellen in IDL***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	IDL	CLOB	☐	☐		*Stellt eine wohlgeformte, semantisch korrekte IDL-Datei dar*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface, IDL

„Component_Interface_Function“	***Auflistung der IDL-Funktionen einer Komponente***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	Name	VarChar	☑	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface_Function

„Component_Interface_DataObject_Term“	***Zuordnung der IDL-Datenobjekte zu Termini der Terminologieebene***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component_Interface_DataObject. Component	
2	NameOfDataObject	VarChar	☑	☑	Component_Interface_DataObject. Name	
3	Term	Int	☐	☑	Term.ID_Term	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface_DataObject, Term

„Component_Interface_Function_Tasks“			***Materialisierte m:n-Beziehung zw. IDL-Funktionen zu Aufgaben der Komponenten***			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Compo-nent	Int	☑	☑	Component_Interface_Function. Component	
2	Name	VarChar	☑	☑	Component_Interface_Function.Name	
3	Task	Int	☑	☑	Component_Tasks. ID_SentenceOfComponent	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface_Function, Compo-nent_Tasks

„Component_Interface_DataObject“			***Auflistung der IDL-Datenobjekte einer Komponente***			
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	Name	VarChar	☑	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Component_Interface_DataObject

Interrelationale Integritätsbedingungen

- ¸ ∀i. Component_Interface_Function[i] ∈ *Functions*(IDL)
- ¸ ∀i. Interface_DataObject[i] ∈ *DataObjects*(IDL)

Objekttypendiagramm siehe Abbildung 65.

Objekttypendiagramme von Abstimmungs-, Verhaltens-, Schnittstellen- und Qualitätsebene

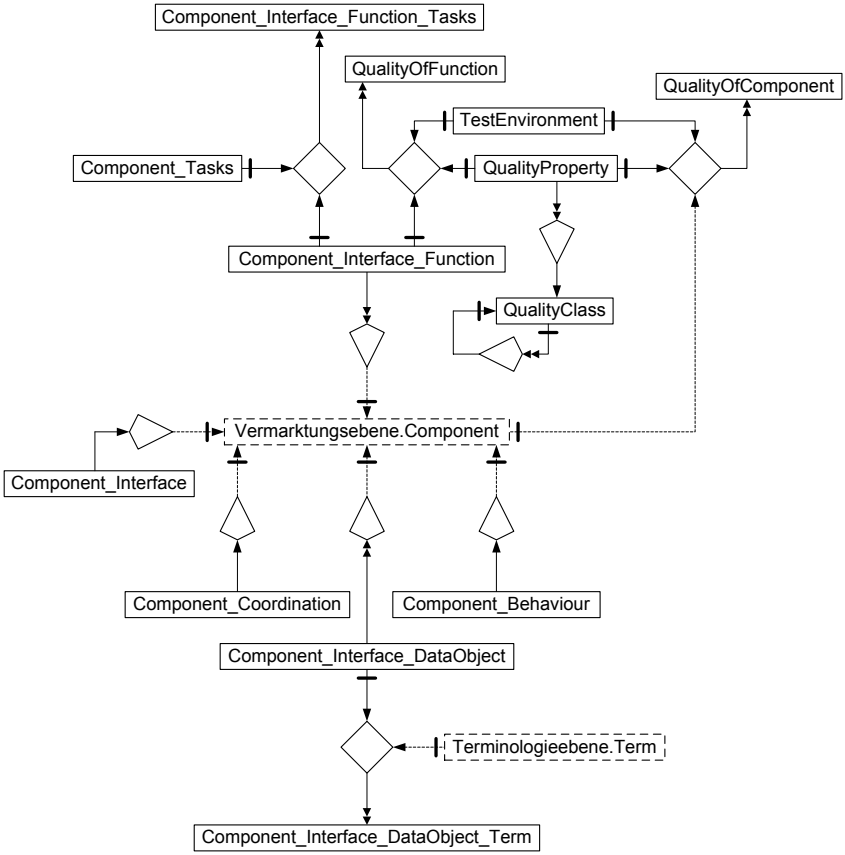

Abbildung 65: Objekttypendiagramm DBKat – Abstimmungs-, Verhaltens-, Schnittstellen- und Qualitätsebene

A.7.1.8 Unterstützung semantischer Stichwortsuchen

Die Unterstützung semantischer Suchen anhand von Stichwörtern erleichtert das gezielte Auffinden von Komponenten insbesondere im Bereich mobiler Endgeräte, welche aufgrund gegebener Geräterestriktionen keine komplexen Suchanfragen auf den im Rahmen von DBKat zur Verfügung gestellten Relationenschemata gestatten. Stichwörter besitzen hierbei eine „Nähe“ zueinander, sodass von eingegebenen Stichwörtern zu „Nachbarn“ (gemäß den im System abgelegten Distanzmaßen) übergegangen werden kann (fuzzy search).

Relationenschemata

„Keyword“	***Speicherung eines Suchschlüsselwortes***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Keyword	Serial	☑	☐		
2	Name	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Keyword

„Keyword_Connection“	***Gewichtete (gerichtete) Netzstruktur zw. den Schlüsselwörtern***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Keyword1	Int	☑	☑	Keyword.ID_Keyword	
2	Keyword2	Int	☑	☑	Keyword.ID_Keyword	
3	Proximity	Real	☐	☐		$0 \leq Proximity \leq 1$
Integritätsbedingungen: ➢ $Keyword1 \neq Keyword2$						

Anmerkungen:

- verwendete Fachbegriffe: Keyword, Keyword_Connection
- „Proximity“ bestimmt die semantische und pragmatische „Nähe“ zweier Schlüsselwörter.
- Synonyme können bspw. dadurch realisiert werden, dass sie symmetrisch mit Proximity „1.0“ verknüpft werden (alle weiteren Verbindungen müssen lediglich eines der Synonyme betreffen).

„Component_Keywords“	*Zuordnung von Schlüsselwörtern zu Komponenten*					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Component	Int	☑	☑	Component.ID_Component	
2	Keyword	Int	☑	☑	Keyword.ID_Keyword	
3	Fit	Real	☐	☐		*0 ≤ Fit ≤ 1*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Component, Keyword
- „Fit“ bestimmt die semantische und pragmatische „Nähe“ eines Schlüsselworts zu einer Komponente.

Es existieren keine interrelationalen Integritätsbedingungen in diesem Abschnitt.

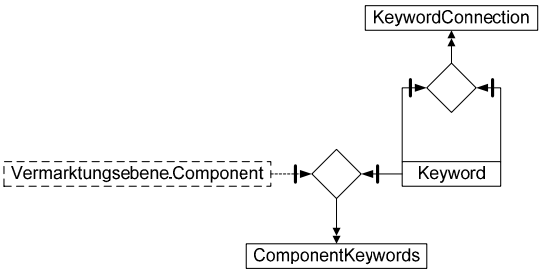

Abbildung 66: Objekttypendiagramm DBKat – Semantische Stichwortsuche

A.7.1.9 Objekttypendiagramm von DBKat

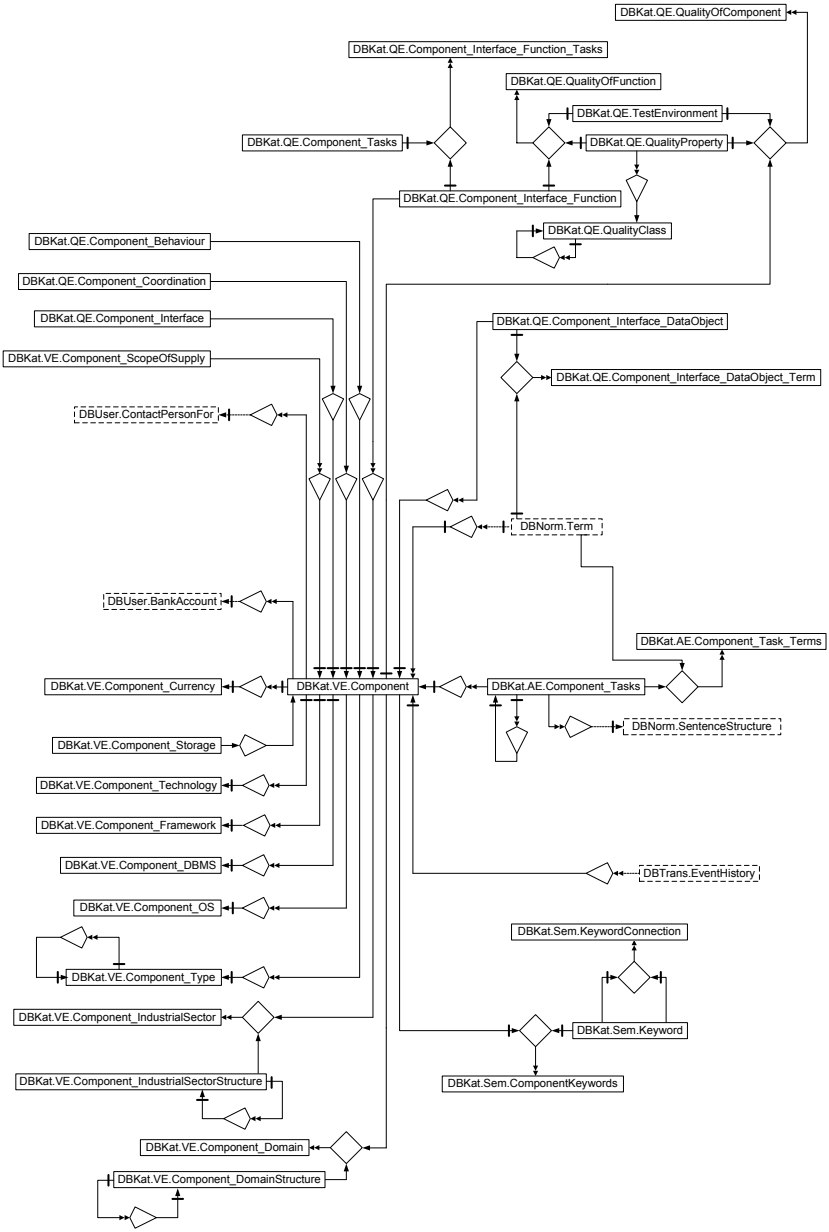

Abbildung 67: Objekttypendiagramm DBKat

A.7.2 DBUser – Benutzer-Datenbank

DBUser speichert sämtliche dem System bekannten natürlichen und juristischen Entitäten strukturiert ab. Hierbei wird unterschieden zwischen natürlichen Personen (Datenstruktur „Person“), welche sowohl als Systembenutzer (verbunden mit einem Log-in-Account) als auch als Ansprechpartner für Unternehmen (Datenstruktur „Contact-Person“ als Resultat der Konnexion mit „Company“) fungieren können und juristischen Personen, welche zur Identifikation des Herstellers der Komponenten dienen. Sowohl Unternehmen als auch Benutzer können über Bankkonten verfügen, deren Daten einheitlich in „BankAccount“ gehalten werden.

Relationenschemata

„Person“	***Hauptstruktur einer natürlichen Person (muss nicht angemeldet sein im System)***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Person	Serial	☑	☐		
2	Salutation	VarChar	☐	☐		*optional*
3	FirstName	VarChar	☐	☐		*optional*
4	MiddleNames	VarChar	☐	☐		*optional*
5	Surname	VarChar	☐	☐		
6	Street	VarChar	☐	☐		*optional*
7	Sex	Char	☐	☐		*optional*
8	ZIP	VarChar	☐	☐		*optional*
9	City	VarChar	☐	☐		*optional*
10	Country	VarChar	☐	☐		*optional*
11	eMail	VarChar	☐	☐		
12	Telephone	VarChar	☐	☐		*optional*
13	Mobile	VarChar	☐	☐		*optional*
14	Fax	VarChar	☐	☐		*optional*
15	Description	CLOB	☐	☐		*optional*
Integritätsbedingungen:						
➢ **¬NULL** ((Street ∧ ZIP ∧ City ∧ Country) ∨ eMail ∨ Telephone ∨ Mobile ∨ Fax)						

Anmerkungen:

- verwendete Fachbegriffe: Person

„LoginAccount“	***Account für den Systemzugang***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_LoginAccount	Serial	☑	☐		
2	Person	Int	☐	☑	Person.ID_Person	
3	LoginName	VarChar	☐	☐		*Unique*
4	LoginPassword	VarChar	☐	☐		*evtl. Sicherheitsanforderungen*
5	BankAccount	Int	☐	☑	BankAccount.ID_BankAccount	
6	IsVendor	Boolean	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: LoginAccount, BankAccount, Vendor

„ContactPersonFor“	***Materialisierte m:n-Beziehung, die Personen Firmen zuordnet***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Company	Int	☑	☑	Company.ID_Company	
2	Person	Int	☑	☑	Person.ID_Person	
3	Position	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: ContactPerson, Company, Person

„Company“	***Herstellerfirmen der Komponenten***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Company	Serial	☑	☐		
2	BankAccount	Int	☐	☑	BankAccount.ID_BankAccount	
3	Name	VarChar	☐	☐		*optional*
4	Street	VarChar	☐	☐		*optional*
5	ZIP	VarChar	☐	☐		*optional*
6	City	VarChar	☐	☐		*optional*
7	Country	VarChar	☐	☐		*optional*
8	eMail	VarChar	☐	☐		
9	Telephone	VarChar	☐	☐		*optional*
10	Mobile	VarChar	☐	☐		*optional*
11	Fax	VarChar	☐	☐		*optional*
12	Description	CLOB	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Company, BankAccount

„BankAccount“	***Kontodaten; Konten werden von Personen oder Firmen besessen***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_BankAccount	Serial	☑	☐		
2	AccountNumber	Int	☐	☐		
3	BankID	Int	☐	☐		
4	BankName	VarChar	☐	☐		
5	IsCreditCard	Boolean	☐	☐		
6	CreditCardNumber	VarChar	☐	☐		
Integritätsbedingungen: ➢ *¬IsCreditCard ∨ (CreditCardNumber ist gültig)*						

Anmerkungen:

- verwendete Fachbegriffe: BankAccount

Es existieren keine interrelationalen Integritätsbedingungen in DBUser.

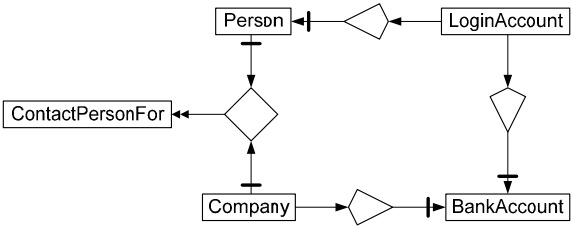

Abbildung 68: Objekttypendiagramm DBUser

A.7.3 DBMeg – Endgeräte-Datenbank

DBMeg dient der strukturierten, hierarchischen Gliederung der vom Marktplatz unterstützten mobilen Endgeräte. Neben Subsumption des jeweiligen Gerätetyps unter eine der hierarchisch gegliederten Geräteklassen werden Informationen abgelegt, die zur Darstellung der Marktplatz-Bedienumgebung auf dem Endgerät sowie der Unterstützungsfunktionalität bezüglich Downloads von anderen Marktplatzkomponenten verwendet wird.

Relationenschemata

„DeviceClass“	***Daten über die Geräteklassen, denen ein oder mehrere Gerätetypen zugeordnet werden***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_DeviceClass	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Description	CLOB	☐	☐		
4	FatherClass	Int	☐	☑	DeviceClass.ID_DeviceClass	
Integritätsbedingungen: ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherClass“ existieren*						

Anmerkungen:

- verwendete Fachbegriffe: DeviceType, DeviceClass
- „FatherClass“ ermöglicht eine hierarchische Strukturierung der Komponententypen.

Es existieren keine interrelationalen Integritätsbedingungen in DBMeg.

„DeviceType“	***Daten über einen einzelnen Gerätetyp***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_DeviceType	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Producer	VarChar	☐	☐		
4	DeviceClass	Int	☐	☑	DeviceClass.ID_DeviceClass	
5	ResolutionX	Int	☐	☐		*optional*
6	ResolutionY	Int	☐	☐		*optional*
7	ColorDepth	Int	☐	☐		*optional*
8	MemorySize	Int	☐	☐		*optional*
9	OS	Int	☐	☑	Component_OS.ID_OS	*optional*
10	WMLSupport	Boolean	☐	☐		
11	HTMLSupport	Boolean	☐	☐		
12	Description	CLOB	☐	☐		*optional*
Integritätsbedingungen:						
➢ *(WMLSupport ∨ HTMLSupport)*						

Anmerkungen:

- verwendete Fachbegriffe: DeviceType, DeviceClass, OS
- Producer hängt nicht mit entsprechenden Herstellertabellen von DBUser und DBKat zusammen.

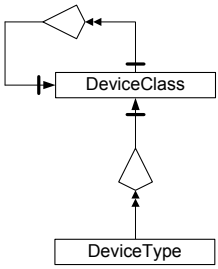

Abbildung 69: Objekttypendiagramm DBMeg

A.7.4 DBTrans – Transaktionen-Datenbank

DBTrans dient als operative Logbuch-Datenbank; jegliche systemrelevanten Ereignisse werden hier chronologisch – zu Debugging-Zwecken, als Datenfundament für Statistiken und für Sicherheitsabfragen (bspw. versuchtes Mehrfach-Einloggen) etc. – protokolliert.

Systemrelevante Ereignisse lassen sich untergliedern in benutzer-, komponenten- und systemorientierte Ereignisse. Benutzerorientierte Ereignisse sind Login und Logout von Benutzern, Registrierung und Abmeldung vom Marktplatzsystem sowie Änderung von Benutzerdaten. Komponentenorientierte Ereignisse sind Auswählen der Detailansicht einer Komponente, Kauf und Download von Komponenten, Einfügen, Ändern und Löschen von Komponenten. Eine strukturierte Gliederung der im Gesamtsystem möglichen Fehlerfälle ist den systemorientierten Ereignissen zu subsumieren (siehe hierzu auch DBError).

Relationenschemata

„EventType“	***Beschreibung der versch. Ereignistypen***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_EventType	Serial	☑	☐		
2	TableName	VarChar	☐	☐		*optional*
3	Description	VarChar	☐	☐		*optional*
Integritätsbedingungen: ➢ *TableName ist entweder NULL oder verweist auf eine im System existente Relation*						

Anmerkungen:

- verwendete Fachbegriffe: EventType

„EventHistory“	***Hauptstruktur zur chronologischen Speicherung relevanter Ereignisse***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Event	Serial	☑	☐		
2	Timestamp	Timestamp	☐	☐		
3	EventType	Int	☐	☐	EventType.ID_EventType	
4	LoginAccount	Int	☐	☑	LoginAccount.ID_LoginAccount	*optional*
5	Component	Int	☐	☑	Component.ID_Component	*optional*
6	UserIPAddress	VarChar	☐	☐		*optional*
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Event, EventHistory, EventType, LoginAccount, Component

„EventType_Login“	***Nähere Informationen zu Login-Ereignissen***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Event	Int	☑	☑	EventHistory.ID_Event	
2	EnteredLoginName	VarChar	☐	☐		
3	EnteredPassword	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Event, EventType
- Eine Tabelle „EventType_Logout“ wäre unnötig, da entsprechende Informationen des korrelierenden Login-Eintrags herangezogen werden können.

„EventType_BuyComponent“	*Kauf von Komponenten*					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Event	Int	☑	☑	Event.ID_Event	
2	BankAccount	Int	☐	☑	BankAccount.ID_BankAccount	
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Event, EventType, Component, BankAccount

„EventType_Error“	*Fehlerereignisse*					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Event	Int	☑	☑	Event.ID_Event	
2	ErrorCode	Int	☐	☑	Errors.ID_Error	
3	ErrorMessage	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Event, EventType, Component, BankAccount

Es existieren keine interrelationalen Integritätsbedingungen in DBTrans.

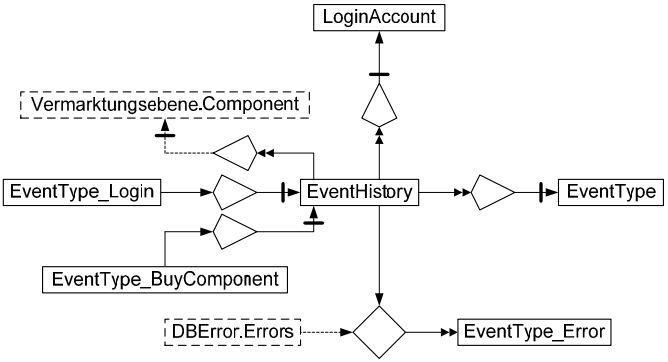

Abbildung 70: Objekttypendiagramm DBTrans

A.7.5 DBMIS – Metainformationssystem

DBMIS stellt Metainformationen, d. h. Informationen über die im System existierenden Schemata (Konzepte wie Relationen, Attribute, Integritätsbedingungen etc.) strukturiert dar. Hierbei erfolgt eine transparente Auflösung von Synonymen auf Attributebene durch den Bezug auf die normsprachliche Daten (DBNorm).

Relationenschemata

„Attribute“	***Metadaten über im System existenten, den Tabellen zugeordneten Attributen***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Attribute	Serial	☑	☐		
2	AttributeName	VarChar	☐	☐		
3	RelationName	VarChar	☐	☑	Relation.ID_Relation	
4	Term	Int	☐	☑	Term.ID_Term	*optional*
5	DataType	VarChar	☐	☐		
6	PartOfPrimaryKey	Boolean	☐	☐		
7	ForeignKey	Int	☐	☑	Attribute.ID_Attribute	
8	Constraint	VarChar	☐	☐		
9	Description	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Attribute, Relation, Term, DataType, PrimaryKey, ForeignKey, Constraint

„Relation“	*Metadaten über im System existente Tabellen*					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Relation	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Term	Int	☐	☑	Term.ID_Term	*optional*
4	Description	VarChar	☐	☐		
5	ExecutiveAdmin	VarChar	☐	☐		
6	CreationDate	Date	☐	☐		
7	DateOfLastBackup	Date	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Relation, Term

Es existieren keine interrelationalen Integritätsbedingungen in DBMIS.

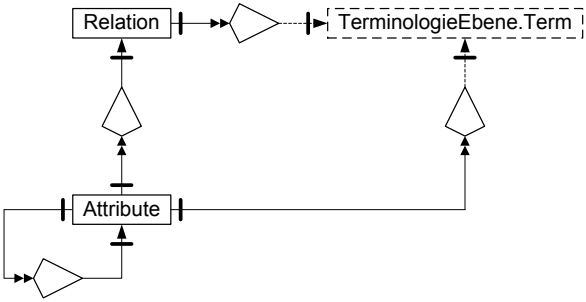

Abbildung 71: Objekttypendiagramm DBMIS

A.7.6 DBNorm – Normsprachenschema

Die Normsprachenebene stellt eine ontologiebasierte Funktionalität für zwei Sprachebenen bereit: Auf der objektsprachlichen Ebene werden Termini und Satzbaupläne zur Beschreibung von im Marktplatz gehandelten Komponenten abgelegt, auf der Metaebene hingegen werden metasprachlichen Konstrukten (Attribute, Relationen) Termini zugeordnet, sodass eine transparente Synonymitätsauflösung ermöglicht wird.

Zur Erreichung dieser Ziele erfolgt eine explizite, schrittweise und zirkelfreie Definition von Fachbegriffen (Termini), welche nicht mit den Schlüsselbegriffen (keywords) zu verwechseln sind: Während erstere in der Aufgabenebene verwendet werden und in der Regel Datenstrukturen von Komponenten normalsprachlich bezeichnen, dienen Schlüsselwörter der semantischen Suche nach Komponenten. Definiert werden Termini nach den gängigen Definitionsregeln – neben obligatorischer Kurzdefinition sind Langdefinitionen, Prädikatorenregeln und Verwendungsbeispiele möglich. Desweiteren können Termini zueinander in Beziehungen (z. B. „ist Gattung von“ oder „ist Schlüssel für“) stehen.

Objektsprachliche Termini können entweder global – dies ist insbesondere zur Standardisierung und der daraus resultierenden Schaffung von Interoperabilität wichtig – oder lokal für einzelne Komponenten gelten. Metasprachliche Termini sind in diesem Sinne stets als global anzusehen.

Relationenschemata

„Term“	***Hauptstruktur zur Definition eines (globalen oder lokalen) Terms***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Term	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	Component	Int	☐	☑	Component.ID_Component	*optional*
4	LanguageLevel	Int	☐	☐		
5	ShortDefinition	VarChar	☐	☐		
Integritätsbedingungen: ➢ *(LanguageLayer = 1 ∨ Component = NULL)*						

Anmerkungen:

- verwendete Fachbegriffe: Term, ShortDefinition, LanguageLevel
- Termini gehören einer bestimmten Sprachebene (LanguageLevel) an
  - „1“ signalisiert die Objektsprachebene und damit die handelbaren Komponenten des Marktplatzes; Termini dieser Ebene können global definiert sein (in diesem Falle besitzt das Feld „Component” den Wert “NULL”) oder zu einer spez. Komponente gehören (in diesem Falle verweist „Component“ auf die entsprechende Komponente).

- „2“ signalisiert die Metasprachebene und damit Strukturen des Marktplatzes (z. B. Relationen, Attribute, Integritätsbedingungen etc.) – Sprachkonstrukte dieser Ebene beziehen sich nicht auf Komponenten und besitzen demgemäß im Feld „Component“ den Wert „NULL“.

„Term_PredicationRules“	***Prädikatorenregeln zur Definitionen der Termini***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Term	Int	☑	☑	Term.ID_Term	
2	ID_PredicationRuleOfTerm	Int	☑	☐		
3	PredicationRule	VarChar	☐	☐		*well-formed formula; die referenzierten Fachbegriffe müssen existieren*
Integritätsbedingungen: ➢ *ID_PredicationRuleOfTerm sollte innerhalb jedes Terminus fortlaufend verwendet werden*						

Anmerkungen:

- verwendete Fachbegriffe: Term, PredicationRule

„Term_ExemplaryUsages“	***Definition eines Terms über Einsatzbeispiele***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Term	Int	☑	☑	Term.ID_Term	
2	ID_ExemplaryUsageOfTerm	Int	☑	☐		
3	ExemplaryUsage	VarChar	☐	☐		
Integritätsbedingungen: ➢ *ID_ExemplaryUsageOfTerm sollte innerhalb jedes Terminus fortlaufend verwendet werden*						

Anmerkungen:

- verwendete Fachbegriffe: Term, ExemplaryUsage

„Term_LongDefinition“		***Ausführliche Definitionen der Termini***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Term	Int	☑	☑	Term.ID_Term	
2	LongDefinition	CLOB	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Term, LongDefinition

„Term_Relationships“		***Beziehungen zw. verschiedenen Termini***				
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	Term1	Int	☑	☑	Term.ID_Term	
2	Term2	Int	☑	☑	Term.ID_Term	
3	Description	VarChar	☐	☐		
Integritätsbedingungen: ➢ $Term1 \neq Term2$						

Anmerkungen:

- verwendete Fachbegriffe: Term, Term_Relationship
- Beziehungen zwischen Termini sind weder zwingend symmetrisch noch antisymmetrisch.
- Beispiele für mögliche Beziehungen sind „ist Spezialfall von“, „ist äquipollent zu“, „ist synonym zu“, „enthält“ etc.

„SentenceStructure“	***Satzbaupläne für die Beschreibung von Komponentenaufgaben***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_SentenceStructure	Serial	☑	☐		
2	RelationType	Int	☐	☑	SentenceStructure_RelationType.ID_RelationType	
3	PropositionalForm	VarChar	☐	☐		*Platzhalter werden durch „_“ gekennzeichnet*
4	NoOfPlaceholders	Int	☐	☐		
Integritätsbedingungen: ➢ *„NoOfPlaceholders“ muss der Anzahl der Platzhalter in „PropositionalForm“ entsprechen*						

Anmerkungen:

- verwendete Fachbegriffe: SentenceStructure, RelationType, PropositionalForm, Placeholder

„SentenceStructure_RelationType“	***Baumstruktur der Satzbauplanarten***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritätsbedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_RelationType	Serial	☑	☐		
2	Name	VarChar	☐	☐		
3	FatherRelationType	Int	☐	☑	SentenceStructure_RelationType.ID_RelationType	*optional*
Integritätsbedingungen: ➢ *es sollte genau ein Tupel ohne Eintrag in „FatherRelationType“ existieren*						

Anmerkungen:

- verwendete Fachbegriffe: RelationType
- „FatherRelationType“ ermöglicht eine hierarchische Strukturierung der Beziehungstypen.

Es existieren keine interrelationalen Integritätsbedingungen in DBNorm.

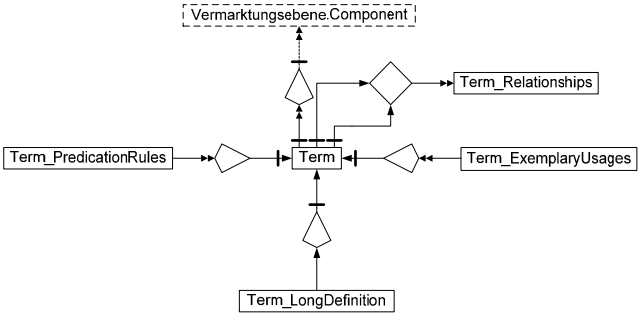

Abbildung 72: Objekttypendiagramm DBNorm – Term

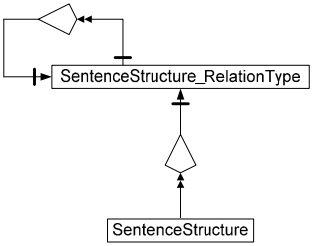

Abbildung 73: Objekttypendiagramm DBNorm – Sentence

A.7.7 DBError – Fehler-Datenbank

Ziel von DBError ist die zentralisierte Administration sämtlicher potenziell möglicher Fehlerereignisse inkl. Gliederung in ihrerseits baumstrukturartig festgehaltenen Fehlerklassen. DBError ist noch als „explorativ“ und somit ausbaufähig einzustufen.

Relationenschemata

„Errors“	***Mögliche Fehlercodes des Systems inkl. Zuordnung zur jeweiligen Fehlerklasse***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_Error	Serial	☑	☐		
2	ErrorClass	Int	☐	☑	ErrorClasses.ID_ErrorClass	
3	Name	VarChar	☐	☐		*Unique*
4	Description	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Error, ErrorClass

„ErrorClasses“	***Hierarchische Strukturierung der möglichen Fehlerklassen***					
#	**Attribut**		**Schlüsseleigenschaften**			**Integritäts-bedingungen**
	Name	Typ	PK	FK		
					Verweis	
1	ID_ErrorClass	Serial	☑	☐		
2	Father	Int	☐	☑	ErrorClasses.ID_ErrorClass	
3	Name	VarChar	☐	☐		*Unique*
4	Description	VarChar	☐	☐		
Integritätsbedingungen: *(keine)*						

Anmerkungen:

- verwendete Fachbegriffe: Error, ErrorClass
- „FatherClass“ ermöglicht eine hierarchische Strukturierung der Fehlerklassen.

Es existieren keine interrelationalen Integritätsbedingungen in DBError.

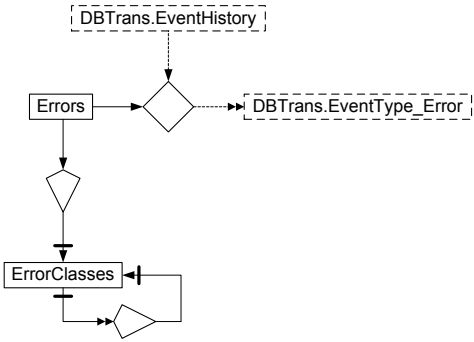

Abbildung 74: Objekttypendiagramm DBError

A.7.8 Gesamtdiagramm

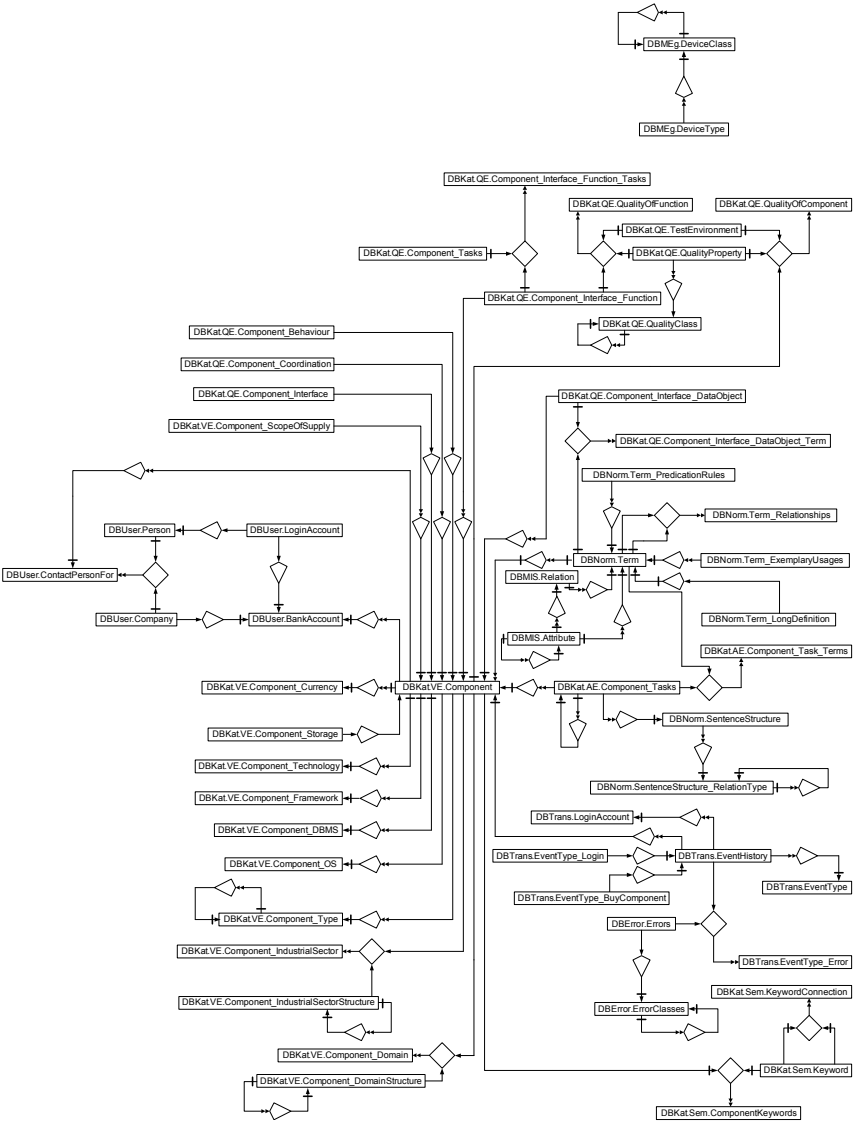

Abbildung 75: Objekttypendiagramm mobiCOMP