Technische Universität München
Institut für Informatik
Lehrstuhl für Wirtschaftsinformatik
Prof. Dr. Helmut Krcmar

Agentenbasiertes Konstruieren von verteilten Systemen am Beispiel Gesundheitswesen

Dissertation

Andreas Schweiger

München, Oktober 2007

Institut für Informatik
der Technischen Universität München

Agentenbasiertes Konstruieren von verteilten Systemen am Beispiel Gesundheitswesen

Andreas A. H. Schweiger

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Johann Schlichter

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Helmut Krcmar
2. Univ.-Prof. Dr. Bernhard Bauer, Universität Augsburg

Die Dissertation wurde am 23.10.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.03.2008 angenommen.

“Apache Tomcat” and “Tomcat” are trademarks of the Apache Software Foundation.

„ARIS“, ARIS Toolset, AML und „IDS“ sind eingetragene Marken der IDS Scheer AG in Deutschland und in vielen anderen Ländern weltweit.

CORBA®, IIOP™, MDA®, Model Driven Architecture®, OMG Interface Definition Language (IDL)™, UML®, Unified Modeling Language™, and Object Management Architecture™ are either registered trademarks or trademarks of Object Management Group, Inc. (OMG™) in the United States and/or other countries.

Der GDT-Inhalt (Gerätedatenträger) ist intellektuelle Ressource des Qualitätsrings Medizinische Software e.V. (QMS).

Diese Software basiert auf den Spezifikationen der Arbeitsgemeinschaft SCIPHOX GbR mbH und dem national adaptierten HL7-Standard der „Clinical Document Architecture (CDA)“. Näheres finden Sie unter http://www.sciphox.de, http://www.hl7.de und http://www.hl7.org.

DOM (Document Object Model), HTML (Hypertext Markup Language), HTTP (Hypertext Transfer Protocol), W3C®, XML (Extensible Markup Language), and XSL (Extensible Stylesheet Language) are registered and unregistered Trademarks of the Massachusetts Institute of Technology (MIT), European Research Consortium for Informatics and Mathematics (ERCIM), or Keio University (Keio).

D2D ist ein eingetragenes Markenzeichen der Kassenärztlichen Vereinigung Nordrhein Körperschaft des öffentlichen Rechts, Düsseldorf.

EJB™, Enterprise JavaBeans™, J2EE™, J2SE™, J2ME™, Java™, Java Card™, JavaBeans™, Java™ Servlets, JDBC™, Java™ RMI, JVM™, K Virtual Machine (KVM), Sun Microsystems, and Solaris™ are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Health Level Seven® and HL7® are registered trademarks of Health Level Seven, Inc. Registered in the U.S. Trademark Office.

IBM®, WebSphere®, and Everyplace® are trademarks of International Business Machines Corporation in the United States, other countries, or both.

IEEE® is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc., (IEEE). This publication is not endorsed or approved by the IEEE.

IrDA® Trademarks are owned by the Infrared Data Association® and used under license therefrom.

i.s.h.med® ist eine eingetragene Marke der GSD Gesellschaft für Systemforschung und Dienstleistungen im Gesundheitswesen mbH in Europa.

JBoss® and Hibernate are registered trademarks and servicemarks of Red Hat, Inc.

In memoriam an meinen Vater Helmut Schweiger.

Zusammenfassung

Die dieser Arbeit zugrunde liegende Vision ist ein Gesundheitswesen, das den Patienten in den Mittelpunkt stellt sowie eine effektive, effiziente und durchgängige Gesundheitsversorgung und Steuerung erlaubt. Dafür bildet die sektor- und institutionsübergreifende Nutzung vernetzter Informationstechnologien eine wesentliche Chance. In dieser Arbeit wird deshalb untersucht, welche besonderen Eigenschaften IT-Systemen zu attribuieren sind, um die genannten Anforderungen zu erfüllen.

Als Ausgangspunkt der vorliegenden Arbeit werden wesentliche Programmiertechniken und damit die zunehmende Abstraktion in der Informatik beleuchtet, um eine geeignete Technologie zur Implementierung für ein IT-System auswählen zu können. Weiterhin werden etablierte Software-Architekturen und insbesondere solche zur Integration von Informationssystemen beschrieben, um wesentliche Eigenschaften für eine geeignete Architektur zur IT-Unterstützung der Informationslogistik zu bestimmen. Eine Literaturanalyse bildet die Basis für die Identifikation des Status quo und von Defiziten im Gesundheitswesen. Als Ergebnisse werden Potenziale dargestellt, die durch IT-Unterstützung induziert werden können und die in einen Kriterienkatalog für ein zukünftiges Informationssystem im Gesundheitswesen einfließen. In einem Anwendungsfall in einem Referenzklinikum werden diese Anforderungen weiter konkretisiert. Dabei wird ein Konzept für ein zukünftiges Informationssystem abgeleitet, welches in einer prototypischen Implementierung umgesetzt wird.

Als Ergebnis der Betrachtung von Programmiertechniken kann gezeigt werden, dass Software-Agenten eine über die Objektorientierung hinausgehende Abstraktion attribuiert werden kann. Weiterhin wird festgestellt, dass sich agentenbasierte Architekturen hinsichtlich ihrer Flexibilität, die eine Voraussetzung für Integrationsszenarien darstellt, auf den Ebenen der IS- und Software-Architekturen besonders auszeichnen: Flexibilität bei IS-Architekturen wird durch ihre Offenheit und Verteiltheit erreicht, die in agentenbasierten Systemen auf der Basis einer standardisierten Plattform gegeben sind. Auf der Ebene der Software-Architekturen wird gezeigt, dass die Flexibilität agentenbasierter Systeme wesentlich durch ihre umfassende Modularisierung sowie ausgeprägte lose Kopplung bestimmt wird. Für die lose Kopplung ist insbesondere die Nachrichtensemantik auf der Basis einer Ontologie relevant. Weiteres Potenzial ist durch eine formale Kommunikationssprache sowie intentionale Informationsmodellierung gegeben. Somit kann zusammenfassend festgestellt werden, dass sich agentenbasierte Architekturen für Domänen unabhängige Integrationsszenarien besonders vorteilhaft einsetzen lassen.

Um eine geeignete Technologie für ein Informationssystem im Gesundheitswesen abzuleiten, wird die Modellierung des Gesundheitswesens als komplexes, adaptives System dargestellt. Es wird gezeigt, dass die Eigenschaften eines solchen Systems in einem agentenbasierten System abgebildet werden können. Mit dieser transitiven Relation ist die Realisierung des Anforderungskatalogs durch ein agentenbasiertes System gegeben. Die durch die getroffenen Architekturentscheidungen gewonnenen Erkenntnisse der prototypischen Implementierung sowie ihre Vorteile werden in Entwicklungsperspektiven für zukünftige Informationssysteme sowie insbesondere einer Darstellung des Mehrwertes durch aktive medizinische Dokumente zusammengefasst.

Abstract

The basis for this dissertation is the vision of future healthcare, which focuses on the patient and allows for effective, efficient, and seamless care and control. The use of networked information technology across sectors and institutions provides essential opportunities to achieve this aim. Therefore, this dissertation examines the attributes which are necessary for IT systems, in order to meet the above mentioned requirements.

Starting with a description of established programming techniques the increasing abstraction in computer science is demonstrated, in order to enable the choice of an adequate implementation technology for an information system. Furthermore, established software architectures and especially those for the integration of information systems are described, in order to determine fundamental characteristics of an adequate architecture for supporting information logistics. A literature analysis forms the basis for the identification of the current status and of deficits in healthcare. As a result, potential areas which could be improved by adequate IT support are accumulated into a requirements list for a future information system in healthcare. This requirements list is substantiated in an application case in a reference clinical centre. Finally, a concept for a future information system is derived, which is instantiated in a prototypical implementation.

As a result of examining programming techniques, one can conclude that software agents can be attributed with a degree of abstraction, that exceeds that of object orientation. Furthermore, agent based architectures are especially characterised by their flexibility in terms of information systems and software architectures. This flexibility is known to be a prerequisite for integration scenarios. Flexibility of information systems architectures is reached by their openness and distribution, which are provided by agent based systems on the basis of a standardised agent platform. Regarding software architectures, flexibility of agent based systems is determined by their comprehensive modularity and excessive loose coupling. In particular, payload semantics and the usage of an ontology in agent based systems contribute to loose coupling. Further potential is provided by a formal communication language and intentional modelling of information. To conclude, agent based architectures can be deployed advantageously for a domain independent implementation of integration scenarios.

In order to argue for an adequate technology for an information system in healthcare, modelling of healthcare as a complex, adaptive system is described. Furthermore, it is demonstrated that the characteristics of such a system can be mapped to an agent based system. Given this transitive relationship, the requirements list can be adequately implemented by an agent based system. The results, which are deduced from the architectural decisions for the prototypical implementation and their advantages, are summarised in perspectives for future information systems in healthcare and in a description of the additional value induced by the construct of active medical documents.

Geleitwort

Der Einsatz von Informationstechnik und Telematik-Technologien im Gesundheitswesen wird bereits seit geraumer Zeit verfolgt. Ziel ist die sektor-, institutions- und prozessübergreifende Nutzung vernetzter Informations- und Kommunikationstechnologien, um horizontal und vertikal konsequent durchgängige, auf die im Zeitlauf veränderliche, individuelle Situation der Patienten abgestimmte Bereitstellung von Gesundheitsdienstleistungen zu ermöglichen. Somit wird ein Gesundheitswesen angestrebt, das mit dem Patienten im Mittelpunkt eine effektive und effiziente Gesundheitsvor- und fürsorge unabhängig von Zeit und Ort erlaubt. Die Eigenschaften des Gesundheitswesens erschweren jedoch die Umsetzung der genannten Ziele.

In der Arbeit von Andreas Schweiger werden auf der Grundlage einer breiten Literaturanalyse aus den Problemstellungen und Gegebenheiten im Gesundheitswesen Anforderungen für ein Informationssystem zur integrierten Versorgung in Form eines Kriterienkatalogs abgeleitet. Hierauf aufbauend wird anhand eines durchgängigen Fallbeispiels in einem Universitätsklinikum eine prototypische Implementierung für ein zukünftiges verteiltes System im Gesundheitswesen erarbeitet. Diese Realisierung basiert auf der Agententechnologie, welche im Vergleich zu etablierten Technologien insbesondere den Mehrwert der Flexibilität besitzt. Agentenbasierten Systemen wird deshalb ein hohes Innovationspotenzial attribuiert. Durch diese Umsetzung kann demonstriert werden, dass solche Systeme ein möglicher Lösungsansatz für die heterogenen Strukturen im Gesundheitswesen sein können.

Die im Rahmen des von der Deutschen Forschungsgemeinschaft geförderten Schwerpunktprogramms 1083 „Intelligente Softwareagenten und betriebswirtschaftliche Einsatzszenarien“ entstandene Arbeit liefert wertvolle Ergebnisse. Letztere wurden von den DFG-Gutachtern als sehr relevant begutachtet, insbesondere weil die Arbeiten Bezug zu der aktuell im Entstehen befindlichen Telematikinfrastruktur für die elektronische Gesundheitskarte nehmen und wertvolle Hinweise für weitergehende Forschungsarbeiten gewonnen werden konnten. Durch die vorliegende Arbeit werden Potenziale beleuchtet, die auf der Grundlage dieser Telematikinfrastruktur in der Form von Mehrwertdiensten gehoben werden können. Aus den Ergebnissen können innovative Ergänzungen für zukünftige Spezifikationen der Telematikinfrastruktur und ihre Komponenten abgeleitet werden.

Vor diesem Hintergrund wünsche ich der Arbeit und den in ihr enthaltenen Lösungsvorschlägen die ihnen gebührende weite Verbreitung.

München, im Oktober 2007 Prof. Dr. Helmut Krcmar

Vorwort

Zum Gelingen der vorliegenden Arbeit trug zu einem wesentlichen Teil mein akademischer Lehrer, Herr Prof. Dr. Helmut Krcmar, bei. Er räumte mir während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an seinem Lehrstuhl für Wirtschaftsinformatik an der Technischen Universität München die Möglichkeit zur Promotion ein. Für seine konstruktiven Anmerkungen und Hinweise sowie die ständige Begleitung bei der Erstellung der Arbeit möchte ich mich sehr herzlich bedanken. Weiterhin schulde ich Herrn Prof. Dr. Bernhard Bauer Dank, der während eines Doktorandenseminars im Rahmen der Konferenz MATES 2005 hilfreiche Anregungen zu meiner Dissertation weitergab und freundlicherweise die Rolle des Zweitgutachters übernahm.

Die Voraussetzungen für die Bearbeitung einer Dissertationsschrift wurden durch meine Eltern Renate und Helmut Schweiger geschaffen. Sie ermöglichten mir durch ihre Unterstützung das Studium der Informatik an der Technischen Universität München. Dafür möchte ich mich sehr herzlich bedanken. Herr Dr. J. S. Fitzgerald, Betreuer meines Individual Projects, das ich während meines Auslandsstudiums an der University of Newcastle upon Tyne bearbeitete, motivierte mich dazu, meine wissenschaftliche Ausbildung nach der Erlangung des Diploms mit einer Promotion zu ergänzen. Deshalb schuf auch er eine Voraussetzung zur Erstellung meiner Dissertationsschrift.

Während der Entstehung der vorliegenden Arbeit wurde ich durch meine Familie unterstützt. Insbesondere in Phasen, in denen das Schreiben Durchhaltevermögen erforderte, erfuhr ich den wesentlichen Rückhalt durch meine Familie. Deshalb möchte ich mich bei Helmut, Renate, Cornelia, Sonja, Armin, Michael, Emanuel, Madeleine, Hannah und Ferdinand sehr herzlich bedanken. Auch mein Mentor, Herr Herbert Brockhaus, motivierte mich, während der Erstellung der Dissertationsschrift meinen Fokus auf diese Arbeit zu richten. Deshalb gilt auch ihm mein besonderer Dank. Herrn Erich Gruber schulde ich für seine Bereitschaft Dank, die Dissertationsschrift Korrektur zu lesen. Weiterhin bedanke ich mich bei Herrn Dr. Jörg Niggemann, Herrn Udo Poth und Frau Helen Dawson, die freundlicherweise Teile der vorliegenden Arbeit Korrektur lasen. Abschließend möchte ich alle Freunde und Bekannte erwähnen, die durch ihre ermutigenden Zusprüche einen nicht zu vernachlässigenden Beitrag zum Gelingen der Arbeit leisteten.

Auch meinen Kollegen Herrn Dr. Jan Marco Leimeister und Herrn Ali Sunyaev möchte ich danken: Sie boten mir stets bei Fragen Unterstützung und zeigten jederzeit Bereitschaft zur kritischen Reflexion. Meinem Kollegen Herrn Dr. Holger Wittges möchte ich für die Diskussion zu den Themen EAI und SOA danken, aus der wertvolle Ideen für die vorliegende Dissertationsschrift entstanden. Weiterhin gilt meinem Kollegen Herrn Matthias Baume mein Dank, der das Projektlogo erstellte.

Wesentliche Grundlagen der vorliegenden Dissertationsschrift wurden im Rahmen von studentischen Arbeiten und von wissenschaftlichen Hilfskräften erarbeitet. Alle diese Arbeiten entstanden unter meiner Betreuung und ständiger Diskussion von Konzepten. Für das Engage-

ment der Studentinnen und Studenten möchte ich mich sehr bedanken. Im Einzelnen wurden von ihnen folgende Leistungen erbracht:

- Analysearbeiten für OP-Prozesse im Klinikum rechts der Isar der Technischen Universität München: *Theodor Bastian* (2005)
- Analyse des Behandlungsprozesses für das kolorektale Karzinom im Klinikum rechts der Isar der Technischen Universität München: *Carina Crosby* (2005)
- Analyse der aktuellen Situation zum Management von mobilen, medizinischen Geräten im Bereich der Anaesthesie im Klinikum rechts der Isar der Technischen Universität München: *Markus Ortmann* und *Martin Böhme* (2007)
- Implementierung der grundlegenden Architektur für die Informationsintegration verteilter Informationssysteme: *Theodor Bastian* (2005)
- Prototypische Abbildung von ausgewählten Sicherheitseigenschaften der Telematikinfrastruktur für die elektronische Gesundheitskarte: *Nicole Haas* (2005a)
- Portierung der virtuellen Patientenakte auf ein ausgewähltes mobiles Endgerät: *Christian Hillebrand* (2006)
- Erweiterung der agentenbasierten Prozesssteuerung von *Harald Meyer* (2004), Persistenzsicherung von Prozessinformationen in einer Datenbank, Implementierung eines Musterprozesses zur Demonstration der Tragfähigkeit der Implementierung: *Steffen Bergmann*, *Andreas Klinger* und *Johannes Winkler* (2006)
- Visualisierung von Prozessbeschreibungen in Form von EPKen: *Viktor Slavtchev* (2007)
- Ankopplung eines Portalsystems an das Konzept der aktiven medizinischen Dokumente: *Mariya Druker* und *Thomas Michelbach* (2006)
- Entwurf und Implementierung einer Benutzeroberfläche für den Prototyp: *Andreja Jaksic* (2006)
- Redesign und Implementierung der Benutzeroberfläche: *Artem Kaprov* (2007)
- Dynamisches Scheduling von Patiententerminen: *Mathias Riedl* (2007)

Die Code-Basis des entwickelten Prototyps entstand durch die Arbeiten der wissenschaftlichern Hilfskräfte am Lehrstuhl für Software Engineering der Universität Potsdam: Alexander Lauert, Harald Meyer und Marco Puhlmann.

Weiterhin flossen in die vorliegende Arbeit die Implementierungsleistungen der folgenden wissenschaftlichen Hilfskräfte des Lehrstuhls für Wirtschaftsinformatik ein: Christian Hillebrand, Johannes Horak und Petromil Petkov. Sie ermöglichten durch ihr Engagement in Form von Diskussionen und Implementierungen die Fertigstellung des beschriebenen Prototyps. Petromil Petkov erstellte darüber hinaus unter meiner Betreuung einige Abbildungen und erarbeitete die Grundlagen für die Installations- und Deployment-Anleitungen der implementierten Applikation. Auch bei Christoph Theel möchte ich mich für seine Unterstützung bei der Erstellung von Abbildungen für die vorliegende Arbeit bedanken.

Schließlich konnte ich bei der Erstellung der Dissertationsschrift von der Integration in das DFG-Schwerpunktprogramm 1083 „Intelligente Softwareagenten und betriebswirtschaftliche

Einsatzszenarien“ profitieren, indem ich mit den darin involvierten Kollegen über Problemstellungen rege diskutieren konnte. Besonderer Dank gilt an dieser Stelle Frau Astrid Hoffmann, Herrn Dr. Stephan Wilczek und Herrn Dr. Thomas Reinke, die an den Forschungsarbeiten im Rahmen des genannten Schwerpunktprogramms wesentlich beteiligt waren und somit auch Voraussetzungen für die Ergebnisse der vorliegenden Arbeit schufen. Diese Resultate entstanden im Rahmen einer Zusammenarbeit im Tandem zwischen dem Lehrstuhl für Wirtschaftsinformatik an der Universität Hohenheim bzw. der Technischen Universität München und dem Lehrstuhl für Software Engineering an der Universität Potsdam. In diesem Teilprojekt „Agentensystemarchitekturen für aktive medizinische Dokumente – Informationslogistik in multikontextuellen Domänen“ (ASA*inlog*) wurde das Konzept der aktiven medizinischen Dokumente erarbeitet, dessen Ideen in der vorliegenden Arbeit aufgegriffen und erweitert wurden. Aus diesem Grund werden auch bei *Reinke* (2003) die Konzepte aktive medizinische Dokumente bzw. bei *Wilczek* (2007) aktive elektronische Dokumente behandelt.

München, im Oktober 2007 Andreas Schweiger

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABAP	Advanced Business Application Programming
ACC	Agent Communication Channel
ADT	Abstract Data Type
ADT	Abrechnungsdatentransfer
AID	Agent Identifier
AMD	Aktives medizinisches Dokument
AML	ARIS Markup Language
AMS	Agent Management System
ANSI	American National Standards Institute
AP	Agent Platform
API	Application Programming Interface
ARB	Agent Resource Broker
ArBaCon	Architecture-Based Construction of Multiagent Systems
ARIS	Architektur integrierter Informationssysteme
AWT	Abstract Window Toolkit
BAPI	Business Application Programming Interface
BDSG	Bundesdatenschutzgesetz
BDT	Behandlungsdatentransfer
bIT4health	better IT for better health
BO	Business Object
CAS	Complex Adaptive System

CCM	CORBA Component Model
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CLI	Common Language Infrastructure
CLR	Common Language Runtime
CLS	Common Language Specification
COM	Component Object Model
CommonKADS	Knowledge Based Systems Analysis and Design Support
CORE	Coding Reference Model
CPW	Clinical Pathway
CTS	Common Type System
D2D	Doctor to Doctor
DCOM	Distributed COM
DF	Directory Facilitator
DHCP	Dynamic Host Configuration Protocol
DOM	Document Object Model
DRG	Diagnosis Related Group
EAV	Entity, Attribute, Value
EDIFACT	Electronic Data Interchange for Administration, Commerce and Transport
eGK	Elektronische Gesundheitskarte
EHIC	European Health Insurance Card
eHIP	eHealth Interoperability Plattform

EII	Enterprise Information Integration
EJB	Enterprise JavaBeans
EPK	Ereignisgesteuerte Prozesskette
FIPA	The Foundation for Intelligent Physical Agents
FPG	Fallpauschalengesetz
GALEN	Generalised Architecture for Languages, Encyclopædias and Nomenclatures in Medicine
GDT	Gerätedatenträger
GEHR	Good European Health Record
GPRS	General Packet Radio Service
GRAIL	GALEN Representation and Integration Language
HBA	Heilberufsausweis
HISA	Healthcare Information Systems Architecture
HL7	Health Level 7
HPC	Health Professional Card
HTTP	Hypertext Transfer Protocol
ICD	International Classification of Diseases
IHE	Integrating the Healthcare Enterprise
IIOP	Internet Inter-Orb Protocol
IMP	Information Module Profile
IMP-NG	Information Module Profile – Next Generation
IrDA	Infrared Data Association
IS	Informationssystem

JAAS	Java Authentication and Authorization Service
JADE	Java Agent Development Framework
JCo	Java Connector
JDBC	Java Database Connectivity
JPEG	Joint Photographic Experts Group
JSR 168	Java Specification Request 168
jUDDI	UDDI in einer Java-Realisierung
JVM	Java Virtual Machine
JXTA	Juxtapose
KBV	Kassenärztliche Bundesvereinigung
KIS	Krankenhausinformationssystem
KV	Kassenärztliche Vereinigung
KVK	Krankenversichertenkarte
KVM	K Virtual Machine
LAN	Local Area Network
LDT	Labordatenträger
LEAP	Lightweight Extensible Agent Plattform
LOINC	Logical Observation Identifiers Names and Codes
MAC	Media Access Control
MAS	Multiagentensystem
MAS-CommonKADS	Multiagent Systems-CommonKADS
MaSE	Multiagent Systems Engineering
MASIF	Mobile Agent System Interoperability Facility

MDA	Model Driven Architecture
MESSAGE	Methodology for Engineering Systems of Software Agents
MIDP	Mobile Information Device Profile
MIME	Multipurpose Internet Mail Extensions
MKT	Multifunktionales Kartenterminal
MRI	Klinikum rechts der Isar der Technischen Universität München
MTP	Message Transport Protocol
MTS	Message Transport Service
MVC	Model, View, Controller
NDA	Needs Driven Approach
ODBC	Open Database Connectivity
OMG	Objekt Management Group
openEHR	open Electronic Health Record
ORB	Object Request Broker
PACS	Picture Archiving and Communication System
PaDok	Patientenbegleitende Dokumentation
PDA	Personal Digital Assistant
PIM	Platform Independent Model
PIN	Persönliche Identifikationsnummer
PML	Process Markup Language
PSM	Platform Specific Model
PVS	Praxisverwaltungssystem

RCTx	Radio-Chemotherapie
RIM	Reference Information Model
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAX	Simple API for XML
SCIPHOX	Standardization of Communication between Information Systems in Physician Offices and Hospitals using XML
SDGO	Stammdatei Gebührenordnung
SDKT	Stammdatenkostenträger
SDRW	Stammdatei Regelwerk
SeSAm	Shell for Simulated Agent Systems
SGB	Sozialgesetzbuch
SICCT	Secure Interoperable ChipCardTerminal
SL	Semantic Language
SMC	Secure Module Card
SMTP	Simple Mail Transfer Protocol
SNOMED CT	the Systematized Nomenclature of Medicine Clinical Terms
SOA	Service orientierte Architektur
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security

TP	Transport Protocol
TTZ	Tumortherapiezentrum
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UMLS	Unified Medical Language System
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
VODD	Verordnungsdatendienst
VPN	Virtual Private Network
VSDD	Versichertenstammdatendienst
WAP	Wireless Application Protocol
WEP	Wired Equivalent Privacy
WLAN	Wireless LAN
WPA	Wi-Fi Protected Access
WSDL	Web Services Description Language
WSIG	JADE Web Service Integration Gateway
XI	Exchange Infrastructure
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations
ZIS	Zugangs- und Integrationsschicht

1 Einführung

1.1 Dimensionen für den Informationsbedarf im Gesundheitswesen

Im Gesundheitswesen können die folgenden beiden Dimensionen für den Informationsbedarf identifiziert werden (siehe dazu auch Roland Berger & Partner GmbH – International Management Consultants 1997, 9-12), wobei Handlungsbedarf zur Beseitigung bestehender Problemfelder wie Qualitätssicherung trotz steigender Patientenzahlen und Reduktion finanzieller Mittel oder die zunehmende Informationsfülle (Roland Berger & Partner GmbH – International Management Consultants 1997, 8) beobachtet werden kann: Gesundheitsverwaltung und medizinische Versorgung (siehe auch Haux et al. 2002, 3 und 17-19). Die Qualität eines Behandlungsprozesses kann aus medizinischer Perspektive trotz der genannten erschwerten Bedingungen erhöht werden, wenn eine elektronische Patientenakte (für die folgenden Ausführungen siehe Abbildung 1.1-1) relevante Informationen vorhält. Weiterhin können die daraus abgeleiteten Daten anonymisiert zur Entscheidung und Steuerung im Gesundheitssystem herangezogen werden. Zentrale Wissensdienste können fallbezogene Informationen für die Behandlung des Patienten zur Verfügung stellen (für die Identifikation dieses Bedarfs siehe auch Porter/Teisberg 2004, 71-72 und 74) und auch als Entscheidungsgrundlage für die Planung im Gesundheitssystem fungieren. Voraussetzung für die Realisierung der genannten Aspekte ist eine umfassende „Gesundheitsplattform“ (Roland Berger & Partner GmbH – International Management Consultants 1997, 12), auf deren Basis geeignete Dienste zur Verfügung gestellt werden.

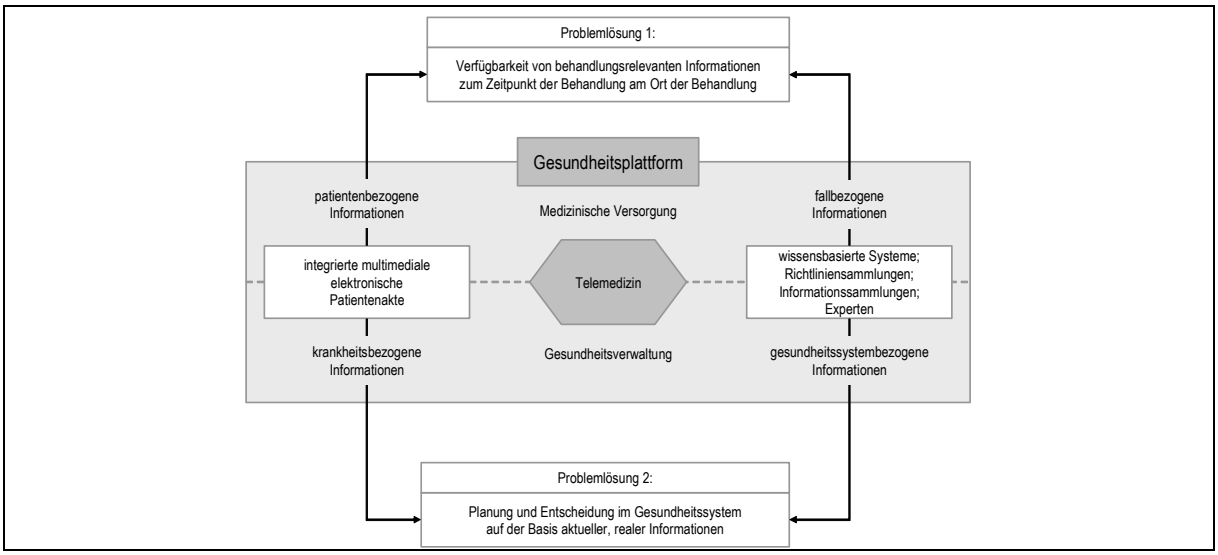

Abbildung 1.1-1: ***Dimensionen für den Informationsbedarf im Gesundheitswesen***
Quelle: *Roland Berger & Partner GmbH – International Management Consultants* (1997, 11)

Die vorliegende Arbeit fokussiert die Konstruktion einer elektronischen Patientenakte, die neben der Bereitstellung von behandlungsrelevanten Informationen gleichzeitig als Grundlage für die Weitergabe des Wissens zu Entscheidungsträgern im Gesundheitssystem fungieren kann. Grundlage dafür ist die im Entstehen befindliche Telematikinfrastruktur (TI) für die

elektronische Gesundheitskarte (eGK, Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005).

1.2 Verteilte Systeme im Gesundheitswesen und ihre Ursachen

Die steigende Anspruchshaltung von Patienten nach einer qualitativ hochwertigen medizinischen Versorgung verschärft den ohnehin existierenden, hohen Kostendruck im Gesundheitswesen. Um diese Anforderungen zu beherrschen, steigt die Notwendigkeit der Einführung von Maßnahmen mit geeignetem Einsparungspotenzial. Dem Einsatz von Informationstechnologie zur Kostenreduktion wird dabei ein hoher Stellenwert beigemessen (Kuhn/Giuse 2001, 275). Die stark ausgeprägte Heterogenität im Gesundheitswesen hinsichtlich der mannigfaltigen Leistungsträger und der damit verbundenen unterschiedlichen Interessen der jeweiligen Gruppen und Verbände sowie eingesetzter Informationssysteme erschwert jedoch die Einführung einer homogenen Lösung.

Ziel der vorliegenden Arbeit ist die Erarbeitung eines Lösungsvorschlags, mit dem im verteilten Gesundheitswesen ein Grundgerüst für eine geeignete IT-Lösung dargestellt wird. Letztere fokussiert eine integrierte Versorgung und Reduktion von informationslogistischen Defiziten unter Beibehaltung bisheriger Informationssysteme und Berücksichtigung der TI für die eGK. Insgesamt wird damit eine Veränderung des Gesundheitswesens hinsichtlich Effizienz, Effektivität, qualitativer Wertigkeit, Transparenz und Kostenreduktion (Graf v.d. Schulenburg et al. 1995, 79-94) angestrebt.

1.3 Motivation und Relevanz

Information hat für die Erbringung der medizinischen Leistung, in besonderer Weise für den Produktionsprozess von Krankenhäusern, eine zentrale Bedeutung (Simoneit 1998, 8). Dabei ergeben sich für die Informationslogistik im Krankenhaus spezielle Anforderungen, deren Erfüllung durch die erforderliche Beherrschung der Dynamik von Prozessen (Kirn et al. 2000, 1) erschwert wird. Dadurch werden die im Gesundheitswesen beobachtbaren signifikanten, informationslogistischen Defizite verstärkt: Probleme im Behandlungsprozess „liegen in unzureichend strukturierten Abläufen, da in vielen Fällen Einzelaufgaben in den Vordergrund gestellt werden und ihr Bezug zum Gesamtkontext vernachlässigt wird“ (Simoneit 1998, 75). Der Grund hierfür liegt darin, dass das Gesundheitswesen durch eine strikte Arbeitsteilung und Spezialisierung bei den Leistungserbringern gekennzeichnet ist. Daraus resultiert die im Gesundheitswesen beobachtbare Verteiltheit.

Im Gesundheitswesen agieren Handlungsträger in unterschiedlichen Rollen. Je nach Rolle ist ein variierender Informationsbedarf zu beobachten. Existierende elektronische Patientenakten sind bezüglich ihres Zugriffs und ihrer Nutzung jedoch lokal eingeschränkt und bieten nicht die Flexibilität in der kontextabhängigen Bereitstellung sowie Auswertung medizinischer und organisatorischer Daten (Paulussen et al. 2003a, 67mm; Kirn et al. 2006a, 203). Da Arbeitsprozesse durch Informationssysteme unterstützt werden, entsteht gleichzeitig eine verteilte Informationssystemlandschaft hinsichtlich Speicherorten und Anwendungen. Diese Aufteilung bedingt die Forderung nach Überbrückung von Schnittstellen und nach Koordination zwischen den am Behandlungsprozess beteiligten Akteuren. Dabei „nimmt eine systematische,

formalisierte Sammlung, Dokumentation und Übermittlung medizinischer und damit zusammenhängender Daten an Bedeutung zu“ (Simoneit 1998, 2). Die relevanten Informationen sind den Handlungsträgern in fehlerfreier, vollständiger, Ziel gerichteter und durchgängiger Form bereitzustellen (Simoneit 1998, 2). Weiterhin ist ein medienbruchfreier, simultaner Zugriff auf Daten durch räumlich verteilte Akteure zu gewährleisten (Simoneit 1998, 2). Ein Lösungsansatz zur integrierten Informationsbereitstellung wird deshalb mit der aktuell im Entstehen befindlichen Telematikrahmenarchitektur (Projektgruppe bIT4health 2004c) und ihrer Lösungsarchitektur für die eGK (Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005) verfolgt.

Die Patientenakte im stationären Bereich ist der zentrale Träger der koordinativen und medizinischen Informationen (Krcmar/Horn 2001b, 3). Einerseits werden medizinische Informationen zur Therapie dokumentiert, andererseits erfolgt die Abstimmung der Arbeitsprozesse zwischen medizinischem und pflegerischem Personal über die Patientenakte (Krcmar/Horn 2001b, 8). Obwohl klinische Informationssysteme mit diesen Anforderungen existieren, bleiben dennoch neben elektronischen oft auch Papierdokumente bestehen (Mikkelsen/Aasly 2001). Als Folge von Medienbrüchen werden Inkonsistenzen in den erfassten Daten beobachtet (Mikkelsen/Aasly 2001). Das Ergebnis der Existenz von Medienbrüchen ist eine fehlerträchtige Übertragung von Daten in das Zielmedium. Folglich können auch fehlerhafte Dokumente identifiziert werden, wobei die potenziellen Konsequenzen einer falschen Information dabei als beträchtlicher eingeschätzt werden als vollständig fehlende Information (Mikkelsen/Aasly 2001).

Fortschritte in der medizinischen Technik und Wissenschaft resultieren in zunehmend komplexen und datenintensiven Behandlungs- und Diagnosemethoden und damit einer Vergrößerung der für den Behandlungsprozess relevanten Informationsmenge. Die zu betrachtende Informationsmenge wächst weiterhin in folgenden zwei Dimensionen (Simoneit 1998, 92-93): Zum einen wird durch die Forschung ein ständig zunehmender Umfang an Informationen produziert. Zum anderen nimmt der Umfang der Informationen in Krankenakten wegen der ansteigenden Arbeitsteilung im Behandlungsprozess zu. Diese Informationsflut verursacht das Problem der mangelnden Deckung des aktuellen, selektiven Informationsbedarfs durch die Gesamtmenge der vorhandenen Informationen.

Obwohl die Kommunikation innerhalb von Institutionen durch die Verbreitung von Kommunikationsstandards wie HL7® (Health Level Seven Inc. Ann Arbor MI 2005) oftmals bereits gegeben ist, stellt der institutsübergreifende Datenaustausch weiterhin eine Hürde dar (Ückert/Müller 2004), weil bislang kein Kommunikationsstandard zum Transport von Daten etabliert ist (Ückert/Müller 2004). Die Vernetzung von Informationssystemen unterschiedlicher Akteure im Gesundheitswesen würde jedoch eine integrierte Informationsversorgung ermöglichen (Ückert/Müller 2004) und damit die Voraussetzung für eine lebenslange Patientenakte bilden, die den Patienten entlang seiner Behandlungen begleitet und nicht nur auf eine bestimmte Institution beschränkt ist.

Diese Entwicklung zur Ausrichtung der Prozesse am Patienten und nicht nur an Abteilungen oder Institutionen berücksichtigt zusätzlich die von den Patienten gewünschte Einsichtnahme in medizinische Dokumente (Fowles et al. 2004; Pyper et al. 2004; Ross et al. 2005). Das Er-

gebnis der genannten Studien bestätigt das Interesse des Großteils der befragten Patienten an der Einsicht in ihre Patientenakten.

Zusammenfassend kann konstatiert werden, dass eine durchgängige Informationslogistik im Gesundheitswesen vermisst wird bzw. noch nicht unterstützt wird. Als globales Ziel zur Beseitigung der identifizierten Defizite wird die Idee der integrierten Versorgung angestrebt, womit vertikal und horizontal durchgängige Prozesse, Daten und Informationstechnologie subsumiert werden. Zu ihrer Realisierung bedarf es der Integration unterschiedlicher Informationssysteme und der vollständigen Digitalisierung von Krankenakten, um das logistische Prinzip für Informationen zu erfüllen. Weiterhin wird als Grundlage eine IT-Infrastruktur vorausgesetzt, die die einzelnen Leistungserbringer miteinander vernetzt. Diese befindet sich aktuell in der TI für die eGK im Entstehen. Der Erreichung des Ziels der integrierten Versorgung stehen aber die beschriebenen Eigenschaften im Gesundheitswesen gegenüber.

Ein weiterer Aspekt ist die Forderung nach der Berücksichtigung der neuen Rolle des Patienten, die diesen als mündigen Bürger identifiziert und ihm die Kontrollmöglichkeit über seine medizinischen Informationen einräumt. Diese Entwicklung, digitale Informationsintegration und Patientenzentrierung, erfolgt meist in mehreren Schritten, wobei dabei fünf Stufen[1] der Digitalisierung (Waegemann 1999) unterschieden werden.

1.4 Forschungsleitende Fragestellungen

Um für die in Abschnitt 1.3 eingeführten Problemstellungen der Informationslogistik im Gesundheitswesen ein geeignetes Lösungskonzept zu erarbeiten, wird die vorliegende Arbeit anhand von forschungsleitenden Fragestellungen strukturiert. Letztere fassen das Forschungsvorhaben in seinen Kernpunkten zusammen und werden in der vorliegenden Arbeit beantwortet:

1. Was sind der Status quo und die Anforderungen im Gesundheitswesen?

Im ersten Schritt werden Problemstellungen in der Informationslogistik beschrieben und der aktuelle Status quo im Gesundheitswesen dargestellt. Weiterhin werden Anforderungen für Informationssysteme im Gesundheitswesen eruiert. Dabei sind insbesondere die Anforderungen der TI für die eGK zu berücksichtigen, auf deren Basis Mehrwertdienste in der vorliegenden Arbeit umgesetzt werden. Die Ergebnisse der Analysearbeit werden in einem Anforderungskatalog verdichtet.

2. Welches Lösungskonzept überwindet die identifizierten Defizite?

Um die gefundenen Defizite zu beseitigen, ist ein geeignetes Konzept zu entwickeln. Es ist dazu insbesondere herauszuarbeiten, welche Technologie bzw. welches Architekturmuster vorteilhaft für die Lösung der aufgezeigten Probleme eingesetzt werden können. Für die Integrationsproblematik von Informationssystemen im Gesundheitswesen ist die Ausrichtung an den Eigenschaften von EAI-Ansätzen (Enterprise Application Integration) sowie Service orientierten Architekturen herauszuarbeiten. Weiterhin ist der Anforderungskatalog aus Forschungsfrage 1 geeignet zu konkretisieren, um für einen betrachteten Behandlungsprozess im Referenzklinikum anhand von ausgewählten Szenarien die Grundlage für das Design der prototypischen Implementierung zu bilden.

3. Welche Elemente und Eigenschaften besitzt eine Architektur für eine prototypische Implementierung, die die Tragfähigkeit des erarbeiteten Lösungskonzepts demonstriert?

[1] Die fünf Stufen der zunehmenden Digitalisierung und Patientenzentrierung werden ausführlich in Abschnitt 4.2.7 beschrieben.

Zur Beantwortung dieser Frage sind prototypische Implementierungen mit ausgewählten Schwerpunkten zu konstruieren. Letztere werden aus dem in Forschungsfrage 2 erarbeiteten Design sowie dem konkretisierten Anforderungskatalog abgeleitet.

4. *Welchen Mehrwert und Nutzen schafft das Konzept des aktiven medizinischen Dokuments?*

Um den Mehrwert und Nutzen der entwickelten Lösung zu zeigen, sind geeignete Evaluierungsmethoden zu identifizieren, um die konstruierten Artefakte zu bewerten. Anhand eines Evaluierungsrahmens sind die Ergebnisse der vorliegenden Arbeit zu evaluieren. Durch einen Vergleich mit einem etablierten Informationssystem im Gesundheitswesen können schließlich Entwicklungspotenziale für zukünftige Informationssysteme erarbeitet werden.

1.5 Wissenschaftstheoretische Betrachtung

Die Disziplin der Wirtschaftsinformatik[2] beschäftigt sich mit „Informations- und Kommunikationssystemen in Wirtschaft und Verwaltung“ (Wissenschaftliche Kommission der Wirtschaftsinformatik (WKWI) 1994, 80). Dabei werden mit dieser Definition zwei verschiedene Aspekte subsumiert:

- Beleuchtung des Zusammenwirkens zwischen der zur Bereitstellung von Information und Kommunikation eingesetzten IT sowie der Organisation (Heinzl/König/Hack 2001, 229)
- Management von IT und Verwendung von IT für Management- und Organisationsaufgaben (Zmud 1997)

Die *Wissenschaftliche Kommission der Wirtschaftsinformatik (WKWI)* definiert Informationssysteme (IS), die Kurzform der Informations- und Kommunikationssysteme, wie folgt (1994, 80):

> **„Bei IS handelt es sich um soziotechnische („Mensch-Maschinen-“) Systeme, die menschliche und maschinelle Komponenten (Teilsysteme) umfassen und zum Ziel der optimalen Bereitstellung von Information und Kommunikation nach wirtschaftlichen Kriterien eingesetzt werden“.**

Die Wirtschaftsinformatik lässt sich so an der Schnittstelle zwischen der realwissenschaftlichen Betriebswirtschaftslehre und der ingenieurwissenschaftlichen Informatik einordnen (Krcmar 2005, 24). Entsprechend dieser Verortung lassen sich auch zwei unterschiedliche wissenschaftliche Ansätze identifizieren: Verhaltenswissenschaftliches und gestaltungsorientiertes Paradigma (March/Smith 1995). Als Schnittstellenwissenschaft bedient sich die Wirtschaftsinformatik beider Ansätze, um insgesamt einen Erkenntnisfortschritt erzielen zu können.

Der verhaltenswissenschaftliche Ansatz hat seinen Ursprung in naturwissenschaftlichen Vorgehensmustern, wobei solche Theorien entwickelt und untermauert werden, die menschliche oder organisatorische Phänomene im Kontext der Analyse, des Designs, der Implementie-

[2] Die Darstellung in diesem Abschnitt ist in Teilen inhaltlich an die Ausführungen bei *Junginger* (2004, 5-7) angelehnt.

rung, des Managements und der Nutzung von Informationssystemen erklären bzw. voraussagen (Hevner et al. 2004, 76). Diese Theorien können über aktuelle Beobachtungen oder ihre Prognosequalität hinsichtlich zukünftiger Beobachtungen begründet werden (March/Smith 1995, 253). Dies ist als Aufgabe der Wissenschaftslogik zu betrachten und wird mit dem Begründungszusammenhang bezeichnet (Behrens 1993, 4770). Der Entdeckungszusammenhang, der Vorgang, der zur Entwicklung der eigentlichen Theorie führt, kann im Gegensatz dazu keiner wissenschaftlichen Überprüfbarkeit unterzogen werden. Entdeckungs- und Begründungszusammenhang werden in Abbildung 1.5-1 (Metamodell für die Forschungsmethode Design Science, siehe dazu auch weiter unten) mit Develop/Build bzw. Justify/Evaluate dargestellt. Ziel des verhaltenswissenschaftlichen Ansatzes ist es, Aussagen über die Effektivität und die Effizienz von Organisationen treffen zu können (Hevner et al. 2004, 76). Im Kontext der verhaltenswissenschaftlichen Betrachtung ist auch die wissenschaftstheoretische Entwicklungslinie der Betriebswirtschaftslehre zu sehen, die vom Empirismus, dem Positivismus und dem Neopositivismus zum kritischen Rationalismus führt (Behrens 1993, 4764-4765). In der deutschen Betriebswirtschaftslehre wurde bereits von *Schmalenbach* (1911-1912) dafür plädiert, die Erklärung des Verhaltens entsprechend dem verhaltenswissenschaftlichen Ansatz mit der Gestaltungsorientierung zu ergänzen. Im Gegensatz zu *Rieger* (1928) wird von *Schmalenbach* gefordert, dass von der Betriebswirtschaftslehre praktisch verwertbares Wissen zu entwickeln ist. Dieser Anspruch wird in der entscheidungsorientierten Betriebswirtschaftslehre auch von *Heinen* (1976) untermauert.

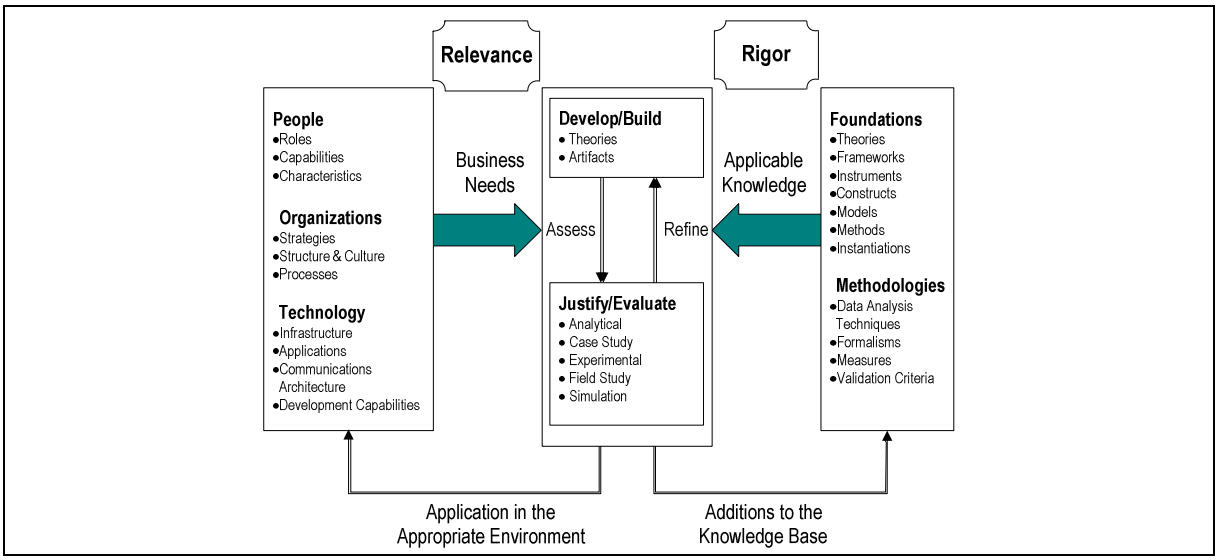

Abbildung 1.5-1: ***Forschungsmethode Design Science***
Quelle: *Hevner et al.* (2004, 80)

Der gestaltungsorientierte Ansatz ist originär der Ingenieurwissenschaft und der artifiziellen Wissenschaft zuzuordnen (Simon 1996). Damit handelt es sich um einen problemlösungsorientierten Ansatz, mit dem Ziel, Innovationen zu generieren, durch die die Analyse, das Design, die Implementierung, das Management und die Nutzung von Informationssystemen effektiv und effizient gestaltet werden können (Hevner et al. 2004, 76). Das in Abbildung 1.5-1 dargestellte Forschungsrahmenwerk für die Wirtschaftsinformatik verknüpft die beiden

komplementären wissenschaftlichen Ansätze (Hevner et al. 2004, 76), um dabei einen Erkenntnisfortschritt erzielen zu können.

Gegenstand der wissenschaftlichen Betrachtung in der Wirtschaftsinformatik stellen Artefakte in unterschiedlicher Ausprägung dar. Differenziert werden dabei nach *Hevner* (2004, 77) Konstrukte (Vokabular und Symbole), Modelle (Abstraktionen und Repräsentationen), Methoden (Algorithmen und Vorgehensmodelle) sowie ihre Instanzen (Implementierungen oder Prototypen). Diese Artefakte sind Objekte der verhaltenswissenschaftlichen Betrachtung, während sich das gestaltungsorientierte Paradigma mit der eigentlichen Konstruktion und Evaluierung der Artefakte beschäftigt (Hevner et al. 2004, 77). Der Beitrag der verhaltenswissenschaftlichen Forschung liegt in der Bewertung der Applikation der Artefakte in der betrachteten Umwelt. Auf diesem Weg können neue Erkenntnisse in die Wissensbasis einfließen (Hevner et al. 2004, 81). Der Gewinn des gestaltungsorientierten Ansatzes liegt hingegen in der innovativen, einzigartigen oder effizienteren, effektiveren Lösung von Problemen (Hevner et al. 2004, 81).

Artefakte können entsprechend den bisherigen Ausführungen als eine Schnittstelle zwischen der inneren (Inner Environment) und der äußeren Welt (Outer Environment) betrachtet werden (Simon 1996, 6). Mit der inneren Welt werden dabei die internen Eigenschaften des Artefaktes, mit der äußeren die Umgebung assoziiert, in der sich das Artefakt befindet und mit der Interaktionen beobachtet werden können. Damit sind die beiden Ansätze zwar verschieden, aber dennoch komplementär, weshalb der verhaltenswissenschaftliche Ansatz dem gestaltungsorientierten gegenüber gestellt werden kann (March/Smith 1995, 252; Simon 1996, 111ff.; Hevner et al. 2004, 76).

Da Artefakte zu einer Problemlösung konstruiert werden, sind sie hinsichtlich der Problemlösungsfähigkeit in ihrer jeweiligen Umgebung zu bewerten. Um die gefundene Lösung im Hinblick auf ihre Optimalität zu evaluieren, können eine Nutzenfunktion als Zielfunktion, Entscheidungsvariablen, Nebenbedingungen und Parameter eingesetzt werden (Simon 1996, 114ff.). Der Umfang der Menge möglicher Alternativen erlaubt meist keine Optimierung, sondern lediglich eine befriedigende Lösung. *Simon* (1996, 119ff.) bezeichnet diesen Sachverhalt mit Satisfizierung.

Die vorliegende Arbeit lässt sich in den Bereich der Gestaltungsorientierung einordnen, da ein Konzept sowie eine Implementierung zur Reduktion von informationslogistischen Defiziten im Gesundheitswesen erarbeitet werden. Obwohl der Fokus im Folgenden auf der Konstruktion von Artefakten zur Unterstützung von Kommunikation und Bereitstellung von Information liegt, kann die vorgeschlagene Lösung im Zusammenhang mit den Ausführungen in Abschnitt 1.1 auch derart betrachtet werden, dass die beiden Aspekte der Wirtschaftsinformatik, d.h. Zusammenwirken zwischen IT und Organisation sowie Management von IT bzw. Verwendung von IT, berücksichtigt werden: IT zur Steuerung des Gesundheitswesens bzw. Unterstützung von Behandlungsprozessen. Die Evaluierung der Ergebnisse (siehe dazu Abschnitt 7) erfolgt in der vorliegenden Arbeit nicht durch eine verhaltenswissenschaftliche Betrachtung, sondern durch die Anwendung von geeigneten Evaluierungsmethoden.

Die vorliegende Arbeit befasst sich mit dem Themengebiet des Informationsmanagements, dessen Ziel die adäquate Verwendung der Information ist (Krcmar 2005, 49). Die Arbeit ist weiterhin im Bereich Gesundheit angesiedelt. Den Ausgangspunkt für die Arbeit bilden die Identifikation von Problemen der Informationslogistik im Gesundheitswesen und die Analyse von Anforderungen (Business Needs in Abbildung 1.5-1) für geeignete Informationssysteme. Die in der Arbeit vorgeschlagenen Problemlösungen werden aus bereits vorhandenem Wissen (Foundations in Abbildung 1.5-1) abgeleitet und durch die erarbeiteten Erkenntnisse ergänzt. Diese Lösungen werden in Artefakten wie Modellen, Instanzbildungen und Implementierungen manifest (Develop/Build in Abbildung 1.5-1). Diese werden evaluiert (Justify/Evaluate in Abbildung 1.5-1), um eine Bewertung der erarbeiteten Ergebnisse vornehmen zu können. Der Erkenntnisgewinn wird in die allgemeine Wissensbasis integriert und kann im Idealfall durch seine Anwendung in der Applikationsdomäne überprüft werden. Auf Letzteres wird in der vorliegenden Arbeit jedoch verzichtet (für eine Begründung siehe Abschnitt 1.11). Die zyklische Entwicklung wird in der vorliegenden Arbeit über ein inkrementelles Vorgehensmodell abgebildet, indem sukzessive Funktionalität hinzugefügt wird. Mit dieser Beschreibung des Vorgehens eignet sich die Forschungsmethode Design Science (Hevner et al. 2004) zur Beantwortung der in Abschnitt 1.4 aufgeführten Forschungsfragen.

Die Methode Design Science (siehe auch Abbildung 1.5-1) wird im Folgenden als Metamethode verstanden, die zur Umsetzung durch Instanzen zu konkretisieren ist: Dafür werden die Vorgehensmodelle Needs Driven Approach (siehe Abschnitt 1.6) zur Orientierung in der Domäne Gesundheitswesen und zur Analyse in Fallstudien, architekturbasierte Konstruktion von Multiagentensystemen (siehe Abschnitt 1.7) zur Entwicklung des zentralen Elements des aktiven medizinischen Dokuments und der inkrementelle Prototypenbau (siehe Abschnitt 1.8) für die Gesamtheit der Implementierung gewählt. Diese Vorgehensmodelle werden in den folgenden Abschnitten 1.6 mit 1.8 beschrieben.

1.6 Needs Driven Approach

Für die Eruierung von beispielhaften Anforderungen in der Domäne Krankenhaus wird der Ethnografie basierte Needs Driven Approach (NDA, Schwabe/Krcmar 1996b, 1996a; Schwabe 2001) eingesetzt, der bereits erfolgreich bei der bedarfsgerechten Gestaltung von Telekooperationssystemen angewandt wurde (Baldi et al. 1996; Gräslund/Krcmar/Schwabe 1996; Schwabe/Hertweck/Krcmar 1997).

In einer Feldstudie zeigte sich, dass die anfängliche Orientierung und Fokussierung auf relevante Aspekte in einem ad hoc unbekannten Feld Problemstellungen mit sich bringen. Der originäre NDA wurde deshalb bei *Schweiger/Krcmar* (2005) um Handlungsanweisungen (siehe Abbildung 1.6-1) ergänzt, die den Modellierer Schritt für Schritt bei der Analyse anleiten (Schweiger/Krcmar 2005, 1013-1014). Diese Handlungsanweisungen sind in mehrere Phasen unterteilt und umfassen fünf Sektionen:

Zunächst wird die Arbeitsumgebung über Räume, Organisationsstruktur und involvierte Akteure beschrieben. Damit wird dem Modellierer eine erste Orientierung für die spätere detaillierte Erfassung der Domäne ermöglicht. Die Ergebnisse dieses Abschnitts werden in Raumplänen, Organigrammen und einer Liste von Akteuren festgehalten.

In Abschnitt zwei folgt die Beschreibung von Aufgaben und Räumen, in denen Akteure diese Aufgaben ausführen. Die Akteure und ihre Aufgaben werden in Anwendungsfalldiagrammen

beschrieben. Indem jeweils eigene Diagramme für verschiedene Örtlichkeiten erstellt werden, werden Akteure mit denjenigen Räumen assoziiert, in denen die Tätigkeiten durchgeführt werden.

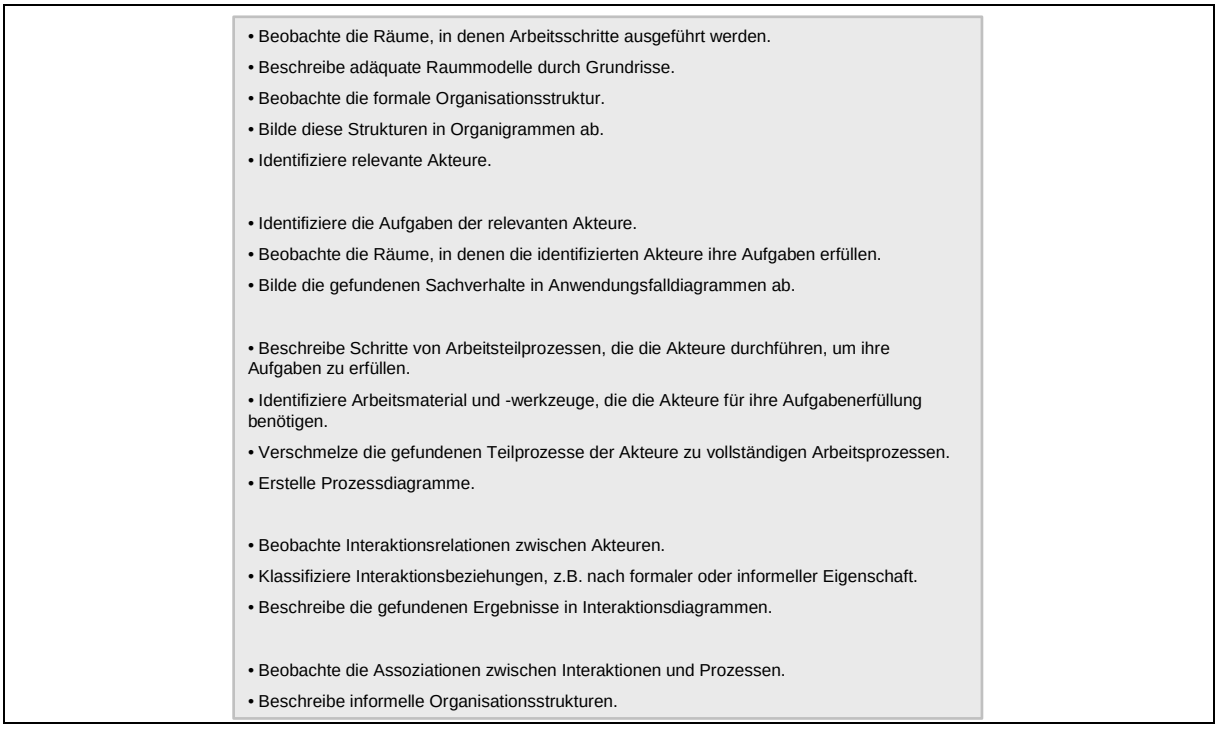

Abbildung 1.6-1: ***Handlungsanweisungen für Analyseschritte beim Einsatz des NDA***
Quelle: *Schweiger* (2005, 1013)

In Abschnitt drei erfolgen die Beobachtung von partiellen Arbeitsabläufen und ihre Integration. Prozessmodelle beschreiben Aktionen und ihre Abhängigkeiten. Akteure, notwendiges Arbeitsmaterial und erforderliche Werkzeuge werden mit diesen Aktionen assoziiert. Schließlich werden die Teilprozesse zu einem Gesamtprozess zusammengeführt. Teile der Interaktionen zwischen Akteuren sind oftmals zu diesem Zeitpunkt bereits bekannt, weil sie durch die Beschreibung der beobachteten Arbeitsprozesse impliziert werden.

In Abschnitt vier werden diese Interaktionen vervollständigt und aus einer anderen Blickrichtung betrachtet, indem sie von den Arbeitsprozessen losgelöst und in separaten Interaktionsdiagrammen beschrieben werden.

Interaktionen und Prozesse sind eng miteinander verbunden. Deshalb werden die Assoziationen zwischen Interaktionen und Prozessen detailliert in Abschnitt fünf beschrieben. Indem Prozesse und Interaktionen voneinander separiert werden, wird die Komplexität der Diagramme reduziert.

Um institutionsübergreifende Prozesse modellieren zu können, kann das NDA-Konzept auf aktive dynamische Handlungsräume (Schweiger/Krcmar 2004) übertragen werden. Ein Handlungsraum bildet dabei einen Zusammenschluss aller zu einem bestimmten Zeitpunkt erforderlichen Informationen, medizinischen Geräte und Akteure ab. Dabei wird insbesondere die

Neubildung von Handlungsräumen berücksichtigt, die z.B. durch die Behandlung eines Patienten bei einem niedergelassenen Arzt und nachfolgend in einem Krankenhaus entsteht. Zur Reduktion der Komplexität wird in der vorliegenden Arbeit auf die Integration dieses Konzepts verzichtet.

Die NDA-Methode wurde bereits wiederholt zur Analyse der Domäne Gesundheitswesen eingesetzt[3]. Dabei zeigte sich, dass dieses Vorgehen adäquat zur Erfassung dieser komplexen Domäne ist. Insbesondere der duale Charakter von Patientenakten zur Dokumentation und Kooperation (Krcmar/Horn 2001b, 3) setzt eine geeignete Methode zur Analyse voraus, welche die Anforderungen für eine Telekooperationsumgebung berücksichtigt (Krcmar/Horn 2001b, 21). Weiterhin eignet sich der NDA mit der Formalisierung in einem objektorientierten Metamodell (siehe dazu die Darstellung der Ergebnisse bei Schweiger/Krcmar 2005, 1015 und 1017) als Ausgangspunkt für eine strukturierte Software-Entwicklung. Aus den genannten Gründen wird der NDA auch in der vorliegenden Arbeit eingesetzt, um die Problemstellungen der Domäne zu erfassen und letztendlich die Konstruktion eines prototypischen Informationssystems zu unterstützen.

1.7 Vorgehensmodell zur Konstruktion einer agentenbasierten Lösung

Etablierte Vorgehensmodelle zur Entwicklung von Multiagentensystemen, wie z.B. MAS-CommonKADS[4] (Iglesias et al. 1997), Gaia[5] (Wooldridge/Jennings/Kinny 1999, 2000), MaSE (Multiagent Systems Engineering, Wood/DeLoach 2000; DeLoach/Wood/Sparkman 2001) oder MESSAGE (Methodology for Engineering Systems of Software Agents, Caire et al. 2001), zeichnen sich zwar durch ihre Aktualität und Akzeptanz in der Wissenschaft (Reinke 2003, 50) aus. Dennoch kann bei diesen Modellen eine Reihe von Eigenschaften identifiziert werden (Reinke 2003, 57), die in jeweils unterschiedlicher Abstufung nicht adäquat unterstützt werden:

- Berücksichtigung von Erkenntnissen über Software-Architekturen
- Theoretische Fundierung der Methoden
- Ganzheitlichkeit der Methoden
- Granularität der Beschreibung von Operationen
- Definition von Iterationszyklen
- Durchgängigkeit der Beschreibung

[3] *Krcmar/Horn* (2001b), *Krcmar/Horn* (2003), *Schweiger/Krcmar* (2005), *Bastian* (2005), *Crosby* (2005).

[4] MAS-CommonKADS ist eine Adaption der Methode CommonKADS (Knowledge Based Systems Analysis and Design Support, Schreiber et al. 1999) zu ihrer Anwendung für Multiagentensysteme (MAS, Multiagent Systems).

[5] *Wooldridge/Jennings/Kinny* (1999, 310) begründen die Wahl der Namensgebung für die von ihnen entwickelte Methode Gaia wie folgt:
„In Greek mythology, Gaia was the mother Earth figure. More pertinently, Gaia is the name of an influential hypothesis put forward by the ecologist James Lovelock, to the effect that all the living organisms on the Earth can be understood as components of a single entity, which regulates the Earth's environment. The theme of many heterogeneous entities acting together to achieve a single goal is a central theme in multi-agent systems research (Bond/Gasser 1988), and was a key consideration in the development of our methodology."

Die Methode ArBaCon (Architecture-Based Construction of Multiagent Systems, Reinke 2003) beschreibt einen Ansatz zur architekturbasierten Entwicklung von Multiagentensystemen, in der die genannten Defizite beseitigt werden. Weiterhin eignet sich diese Methode insbesondere zur Modellierung der gegebenen Domäne Gesundheitswesen (Schweiger/Krcmar 2006, 309-310). Dies wird durch die Beschreibung der ArBaCon-Methode anhand eines durchgängigen Beispiels aus dem genannten Bereich (Reinke 2003) untermauert. Weiterhin kann in diesem Vorgehensmodell der sozialwissenschaftliche Ansätze des NDA (siehe dazu Abschnitt 1.6) mit ingenieurwissenschaftlichen Methoden zur Erstellung eines agentenbasierten Systems verknüpft werden (Krcmar/Horn 2003, 28). Aus diesen Gründen wird für die prototypische Implementierung der vorliegenden Arbeit die ArBaCon-Methode gewählt. Mit dieser wird der Abschnitt Develop/Build aus der Forschungsmethode Design Science (siehe Abbildung 1.5-1) in einer Instanz konkretisiert. ArBaCon ist in Abbildung 1.7-1 dargestellt und besitzt die folgenden Eigenschaften (Reinke 2003, 58-60):

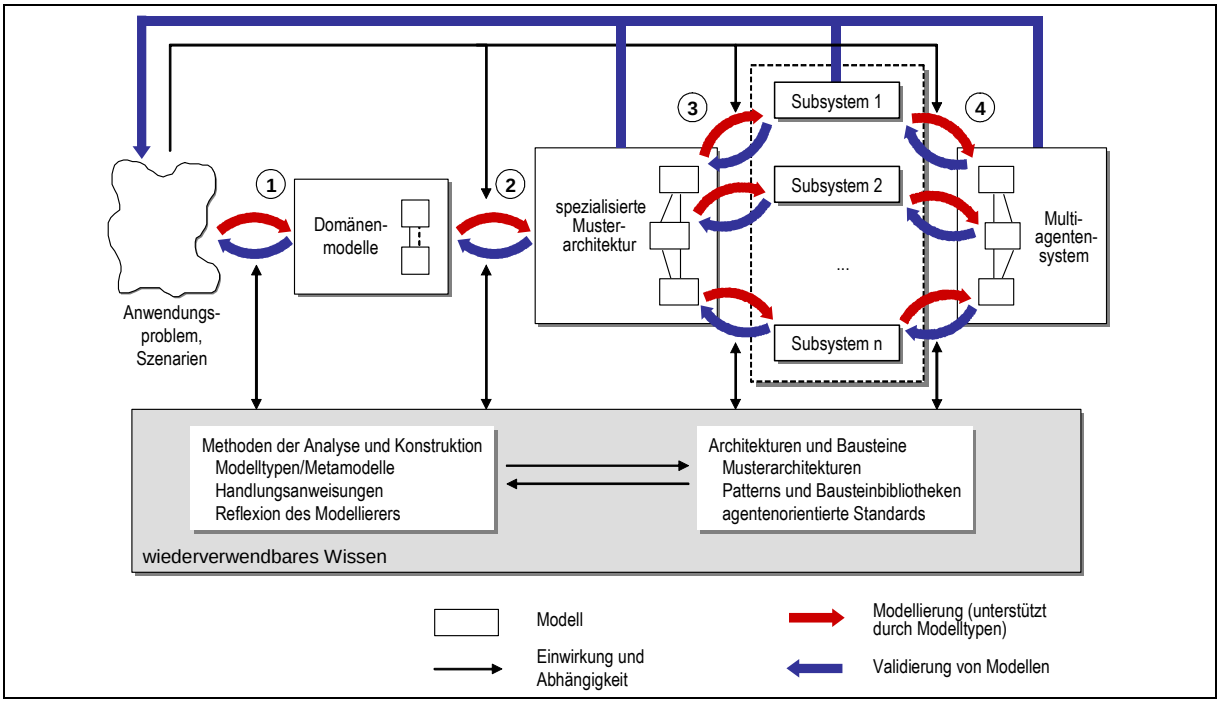

Abbildung 1.7-1: ***Überblick über die Methode ArBaCon***
Quelle: *Reinke* (2003, 59) und in Anlehnung an *Schweiger/Krcmar* (2006, 310)

Das Vorgehensmodell ist in vier Etappen untergliedert, deren Ergebnisse entsprechend einer evolutionären Entwicklung in mehreren Phasen (Analyse, Spezifikation, Entwurf, Implementierung, Betrieb) erarbeitet werden. Weiterhin werden bei diesen Etappen jeweils Eingaben und Ergebnisse in Form von Modellen eingesetzt bzw. generiert. Die erzielten Resultate der Etappen werden gegen die Eigenschaften der Domäne und Modelle aus der vorgelagerten Etappe validiert.

Alle Etappen zeichnen sich durch die Wiederverwendung von Wissen und Produkten aus: Für die jeweils zu konstruierenden Modelle werden von ArBaCon geeignete Modelltypen bzw. Metamodelle beschrieben. Diese werden durch konkrete Handlungsanweisungen ergänzt, die

den Entwickler dazu befähigen, relevante Schritte umzusetzen. Die ständige Reflexion erlaubt die permanente Konsolidierung von Erfahrungswerten über das Wissen und die Produkte. Diese Erfahrungswerte können in den weiteren Verlauf des Entwicklungsprozesses einfließen. Produkte können in Form von Musterarchitekturen zur Beschreibung von agentenbasierten Software-Systemen mit Elementen sowie ihrer Assoziationen, Entwurfsmustern als etablierte Lösungsmöglichkeiten für wiederkehrende Problemstellungen und agentenorientierten Standards z.B. für die zugrunde liegende Agentenplattform oder Kommunikationsmechanismen im Entwicklungsprozess eingesetzt werden. Bei der Konstruktion dieser Produkte wird jeweils das bisher gewonnene Wissen angewendet.

Die weiter oben bereits genannten Etappen des ArBaCon-Vorgehensmodells (für eine ausführliche Beschreibung siehe die Darstellung bei Reinke 2003, 58-162) werden im Folgenden kurz betrachtet: Ausgangspunkt für den Entwicklungsprozess ist das Anwendungsproblem aus einer dedizierten Domäne. Die daraus abgeleiteten Anforderungen fließen in den weiteren Entwicklungsprozess ein und dienen auch zur Validierung der Etappenergebnisse. Diese Resultate werden in den folgenden Schritten erzielt (Reinke 2003, 59-60):

- *Etappe 1 (Domänenmodellierung)*: Ausgehend von den Anforderungen des zu konstruierenden Systems werden relevante Domänenmodelle mit unterschiedlichen Schwerpunkten erstellt (Reinke 2003, 59): Organisationsmodelle (Handlungsträger und ihre Beziehungen), Geschäftsobjektmodelle (z.B. Nachrichteninhalte aus Interaktionen) sowie Prozessmodelle (Aufgaben, ihre Dekomposition und Abläufe). Organisationsmodelle werden dabei in Form von Anwendungsfalldiagrammen, Organisationsstrukturen und Lokationsmodellen dargestellt. In Anwendungsfalldiagrammen werden in Anlehnung an die objektorientierte Analyse Akteure identifiziert, die mit dem zu konstruierenden System oder anderen Akteuren interagieren. Organisationen sowie ihnen zugeordnete Akteure werden in einem erweiterten Klassendiagramm beschrieben. Darin werden insbesondere Assoziationen und ihre Kardinalitäten zwischen den dargestellten Elementen spezifiziert. Im Lokationsmodell werden die räumlichen Gegebenheiten erfasst. Diese Modellierung reflektiert relevante Lokationen sowie ihre assoziierten Organisationseinheiten und Akteure. Die Relationen zwischen den Lokationen werden durch die Angabe von Kardinalitäten näher spezifiziert. Die Modellierung von relevanten Geschäftsobjekten durch Klassendiagramme schließt auch die Erfassung der bereits bestehenden Infrastruktur der Informationssysteme ein. In Prozessmodellen werden Aufgaben identifiziert, ggf. dekomponiert sowie mit den zugehörigen Akteuren und Geschäftsobjekten assoziiert.
- *Etappe 2 (Auswahl und Spezialisierung einer Musterarchitektur)*: Aus einer allgemeinen oder domänenspezifischen Muterarchitektur wird eine spezialisierte Architektur abgeleitet, die die Anforderungen des zu konstruierenden Systems erfüllt. Grundlage dafür sind mehrere Schritte: Zunächst wird die Abbildung der zuvor in der Domänenmodellierung identifizierten Akteure, Ressourcen und Umgebungen auf Autoritäten, Ressourcen und Agentenplattformen bestimmt (siehe auch Reinke 2003, 92). Die in der Prozessmodellierung identifizierten Aufgaben werden in von Agenten ausführbare Aktionen klassifiziert und so geeigneten Typen zugeordnet. Weiterhin werden die bestimmten Agentenaufgaben gruppiert und mit Agentenklassen assoziiert, die für deren Ausführung geeignet sind. Damit werden die Aufgaben entsprechend der gewählten Musterarchitektur Agenten aus einem adäquaten Teilsystem der Gesamtarchitektur zugeordnet. Auf der Basis der identifizierten Agenten und Ressourcen werden die Interaktionen zwischen

diesen bestimmt und in Sequenzdiagrammen dargestellt. Weiterhin werden Konnektoren als Verbindungselemente zwischen Agenten den Interaktionen und ihren Typen zugeordnet. Abschließend wird aus den so gewonnenen Modellen eine spezialisierte Musterarchitektur konstruiert sowie ihr Modelltyp determiniert.

- *Etappe 3 (Konstruktion der Teilsysteme)*: Die spezialisierte Musterarchitektur wird in dieser Etappe in ihren Subsystemen spezifiziert, entworfen, implementiert und getestet. Dabei wird zwischen der Konstruktion der Architekturelemente Agentenklassen sowie Agentenkonnektoren und ihre Abbildung auf technische Modelle differenziert (Reinke 2003, 113-134). Ausgangspunkt für die Konstruktion ist die Verfeinerung von Interaktionen zwischen Agenten, wobei dabei auch Nachrichteninhalte und Adressaten berücksichtigt werden. Weiterhin werden das Verhalten von Agenten in Aktivitätsdiagrammen definiert sowie ihre Attribute und Operationen bestimmt. Abschließend werden in dieser Etappe die Benutzerschnittstelle (Reinke 2003, 137-139) konstruiert sowie die Integration von Ressourcen (Reinke 2003, 139-140) vorgenommen.
- *Etappe 4: (Integration und Allokation)*: In der abschließenden Etappe werden die konstruierten Teilsysteme integriert und die Schnittstellen zwischen den Komponenten der Teilsysteme implementiert. Weiterhin werden die einzelnen Teilsysteme auf ihren Plattformen installiert. Dazu werden den Tasks der Agenten Ausführungslokationen zugewiesen, Mobilitätseigenschaften von Agenten bestimmt, verteilbare Einheiten determiniert, ihre Abhängigkeiten bestimmt und schließlich die Agenten entsprechend geeigneten Kriterien auf Agentenplattformen verteilt (Reinke 2003, 150-161).

Ein in der ArBaCon-Methode besonders herausgearbeiteter Aspekt ist das Konzept der Konnektoren. *Reinke* definiert diese als „Bausteine zur Vermittlung von Interaktionen in Multiagentensystemen“ (2003, 23). Da dieser Baustein aber kein Bestandteil des etablierten FIPA-Standards (The Foundation for Intelligent Physical Agents (FIPA) 2005) ist und demnach in der weit verbreiteten Agentenplattform JADE (siehe dazu Abschnitt 6.3.2) nicht unterstützt wird, wird in der vorliegenden Arbeit auf die explizite Berücksichtigung von Konnektoren verzichtet.

1.8 Prototypenbau

Das Ziel der Entwicklung eines Prototyps ist die Validierung möglicher Anforderungen (Riddle 1984, 25) an ein Software-System. Deshalb ist dabei ein wesentliches Element die Intention, den Prototyp als Lernmedium zu verwenden, um damit das zu entwickelnde System präzise spezifizieren zu können (Floyd 1984, 3). In der vorliegenden Arbeit sind die Anforderungen für ein zukünftiges verteiltes System im Gesundheitswesen zu determinieren. Somit eignet sich der Ansatz des Prototypenbaus.

Obwohl die Konstruktion von Prototypen dann besonders vorteilhaft ist, wenn die Anforderungen durch die Anwender a priori unbestimmt sind (Riddle 1984, 20), wurde dennoch für die in der vorliegenden Arbeit angestrebte Implementierung das inkrementelle Vorgehensmodell des Prototypenbaus gewählt. Der Prozess des Prototypenbaus wird nach *Floyd* (1984, 4) in vier Schritte unterteilt. In den folgenden Ausführungen werden diese Schritte mit Aspekten der vorliegenden Implementierung assoziiert, mit denen die Wahl des inkrementellen Vorgehensmodells für die vorliegende Arbeit begründet wird:

- Die *Auswahl der Funktionalität* des vorliegenden Prototyps ist vertikal, weil nur ausgewählte Funktionen implementiert sind. Die Funktionalität ist zudem horizontal, weil Teile simuliert sind.
- Die *Konstruktion eines Prototyps* umfasst in der Regel einen geringeren Aufwand als die Implementierung des vollständigen Zielsystems (Floyd 1984, 4). Deshalb erfolgt beim Prototypenbau die Konzentration auf diejenigen Funktionalitäten, die dem Zweck des Prototyps entsprechen. Die vorliegende Implementierung zielt auf die Demonstration der Potenziale des Einsatzes von Agentensystemen im Gesundheitswesen ab. Dazu werden nur ausgewählte Einsatzszenarien, medizinische Standards, zu integrierende Informationssysteme, Eigenschaften der TI für die eGK, exemplarische Sicherheitseigenschaften im Prototyp, Elemente eines Patientenportals, Möglichkeiten von mobilen Anwendungen, Komponenten der Prozesssteuerung sowie Funktionalitäten einer Scheduling-Applikation abgebildet.
- In der *Evaluierung* werden die zur Implementierung ausgewählten Funktionalitäten gegen den Anforderungskatalog bewertet. Um die Ergebnisse der Implementierung bewerten zu können, müsste der Anforderungskatalog mit Maßzahlen belegt werden (siehe dazu Riddle 1984, 24-25), um intersubjektiv nachvollziehbare Resultate erzielen zu können. Weil sich dies im Fall von Agentensystemen schwierig gestaltet (siehe dazu die Ausführungen in Abschnitt 1.9), wird in der vorliegenden Arbeit eine qualitative Bewertung vorgenommen. Weiterhin können in dieser Phase des Prototypenbaus Problemstellungen identifiziert werden, die in einen folgenden Iterationszyklus einfließen können. Im vorliegenden Fall konnten z.B. Grenzen bei der erforderlichen Funktionalität der Plattform Java™ auf dem ausgewählten mobilen Endgerät identifiziert werden. Als Konsequenz wurde eine Alternative für die Unterstützung der Java-Laufzeitumgebung ausgewählt.
- Die weitere *Verwendung des Prototyps* zielt auf die Evaluierung des implementierten Systems hinsichtlich Entwicklungsperspektiven zukünftiger Informationssysteme und auf die Gewinnung von neuen Erkenntnissen bezüglich der Eignung von Software-Agenten im Gesundheitswesen ab.

Für den Prototypenbau werden bei *Floyd* (1984, 6) differenzierte Ansätze unterschieden. Von diesen Möglichkeiten des explorativen, experimentellen sowie evolutionären Prototypenbaus (Floyd 1984, 6) ist der evolutionäre Stil (Floyd 1984, 10-12) für das beschriebene Vorhaben geeignet. Dabei werden Anforderungen an das System schrittweise von Version zu Version umgesetzt und die Phasen orientierte Abfolge der Entwicklungsschritte durch das Durchlaufen mehrerer Zyklen ersetzt. Als Vorgehensmodell für den evolutionären Stil wird für die vorliegende Implementierung die inkrementelle Systementwicklung (Floyd 1984, 11) eingesetzt. Dabei wird in mehreren Zyklen durch das Design, die Implementierung und die Evaluierung iteriert, um die verschiedenen Teile der prototypischen Implementierung wie die stationäre Anwendung, die mobile Portierung, die Prozessunterstützung, die Anbindung an ein Patientenportal, die Berücksichtigung von Eigenschaften der TI und die Scheduling-Applikation umzusetzen. Um die Komplexität des zu konstruierenden Systems zu beherrschen, werden pro Iterationszyklus nur Teile der Funktionalität implementiert.

Als Werkzeuge für den Prototypenbau werden bei *Floyd* (1984, 12) das modulare Design, Dialogdesign und Simulation identifiziert. Einzelne Module kapseln dabei Implementierungen über definierte Schnittstellen. Module können damit ohne Seiteneffekte gegen weiterent-

wickelte Implementierungen ausgetauscht werden, falls die Schnittstellen unverändert bleiben. In der vorliegenden Implementierung werden einzelne Module eingesetzt, z.B. der simulierte Kartenleser oder ein Modul bei der Portierung auf ein mobiles Endgerät. Das Dialogdesign wurde in der Implementierung evolutionär durchgeführt. Eine anfänglich rudimentäre Benutzeroberfläche bei *Bastian* (2005) wurde iterativ (Jaksic 2006) durch eine anwenderfreundliche Realisierung (Kaprov 2007) ersetzt. Die grundlegende Funktionalität des eigentlichen Prototyps wurde mit diesem Iterationsschritt nicht erweitert, lediglich die Bedienung des Systems wurde konsolidiert. Das Werkzeug der Simulation wurde deshalb verwendet, da z.B. die Implementierung einer vollständigen Schnittstelle zum ausgewählten Krankenhausinformationssystem zu umfangreich wäre und auch über das Ziel der Demonstration der Tragfähigkeit der beschriebenen Implementierung hinausgehen würde. Die Schnittstelle zum Krankenhausinformationssystem wird deshalb nur in Teilen realisiert. Ein weiteres Beispiel für die Simulation ist die Integration des in der TI für die eGK als wesentliches Element geltenden Kartenlesers. Diese Funktionalität wird zwar als für den Prototyp bedeutsam eingestuft, ihre Simulation ist aber mit dem Ziel der Umsetzung von ausgewählten Anforderungen ausreichend. Die in der Implementierung eingesetzte Modularisierung ist eine wesentliche Voraussetzung für die Simulation von Teilen des Systems. Simulierte Module können bei einer wohl definierten Schnittstelle ohne Seiteneffekte gegen die Module mit vollständiger Implementierung ersetzt werden und so die Weiterentwicklung der Implementierung unterstützen.

1.9 Evaluierung

Ein wesentlicher Bestandteil der Methode Design Science ist die Evaluierung (Hevner et al. 2004, 85). *Hevner et al.* (2004, 86) schlagen dafür die in Tabelle 1.9-1 zusammengefassten Methoden vor. In der Arbeit werden folgende Methoden eingesetzt, die für die Überprüfung der jeweiligen Ergebnisse geeignet sind:

- *Sachkundiges Argument*: Erarbeitung eines Kriterienkatalogs für den Vergleich von agentenbasierten Systemen bzw. der prototypischen Implementierung mit den identifizierten Anforderungen für Informationssysteme im Gesundheitswesen
- *Sachkundiges Argument*: Aufbau einer Argumentationslinie für die Begründung der gewählten Architektur
- *Architekturanalyse*: Demonstration der Eignung der Architektur zur Lösung von identifizierten Problemstellungen über die Darstellung von Implementierungsentscheidungen
- *Szenarien*: Darstellung der Problemlösungsfähigkeit in ausgewählten Szenarien anhand eines konkreten Behandlungsprozesses im Referenzklinikum
- *Sachkundiges Argument*: In dem Prototyp werden Lösungen für ausgewählte Probleme der Informationslogistik im Gesundheitswesen implementiert. Dabei ist der Mehrwert gegenüber bisher erarbeiteten Lösungen zu zeigen. Ein Vergleich mit einer Referenzimplementierung im Gesundheitswesen zeigt das Potenzial von aktiven medizinischen Dokumenten bezüglich der Bereitstellung von Informationen nach dem informationslogistischen Prinzip sowie Weiterentwicklungsmöglichkeiten für zukünftige Informationssysteme im Gesundheitswesen.

- *Testen*: Die in Tabelle 1.9-1 genannten Methoden (Black Box- und White Box-Testen) sind für die Rahmenbedingungen der vorliegenden Arbeit geeignet anzupassen (siehe dazu die Ausführungen in Abschnitt 7.4).

Methode	Erläuterung
Beobachtung	• Fallstudie: Das Artefakt wird in seiner Zielumgebung ausführlich betrachtet. • Feldstudie: Das Artefakt wird in multiplen Projekten hinsichtlich seiner Nutzung betrachtet.
Analyse	• Statische Analyse: Untersuchung des Artefakts hinsichtlich statischer Eigenschaften (z.B. Komplexität) • Architekturanalyse: Überprüfung der Passgenauigkeit des Artefakts hinsichtlich der technischen Informationssystemarchitektur • Optimierung: Demonstration der inhärent optimalen Eigenschaften des Artefakts oder Angabe von Optimalitätsgrenzen hinsichtlich des Verhaltens des Artefakts • Dynamische Analyse: Untersuchung des Artefakts hinsichtlich dynamischer Eigenschaften (z.B. Performanz)
Experiment	• Kontrolliertes Experiment: Untersuchung des Artefakts in einer kontrollierten Umgebung (z.B. Bedienbarkeit) • Simulation: Belegung des Artefakts mit künstlichen Daten
Testen	• Funktionales (Black Box) Testen: Ausführen der Schnittstellen zur Identifikation von Fehlern • Strukturelles (White Box) Testen: Abdeckungstests z.B. hinsichtlich von Ausführungspfaden
Deskription	• Sachkundige Argumentation: Begründung der Nützlichkeit des Artefakts über die Wissensbasis (z.B. relevante Forschung) • Szenarien: Konstruktion von detaillierten Szenarien, um die Nützlichkeit des Artefakts zu demonstrieren

Tabelle 1.9-1: ***Evaluierungsmethoden für Artefakte***
Quelle: In Anlehnung an *Hevner et al.* (2004, 86)

Die Evaluierung von Multiagentensystemen kann insgesamt als schwierig eingeschätzt werden, weil keine allgemein akzeptierten Metriken zum Maß der architektonischen Integrität und der Performanz des implementierten Systems existieren (Shehory/Sturm 2001; Koutkias/Chouvarda/Maglaveras 2005, 536). Weiterhin wurde bis dato kein formales Vorgehensmodell zur Evaluierung von Multiagentensystemen standardisiert, welches die eingesetzte agentenorientierte Methode des Software Engineerings und die gewählte Agentenplattform bewertet (Koutkias/Chouvarda/Maglaveras 2005, 536). Deshalb wurde z.B. bei *Koutkias/Chouvarda/Maglaveras* (2005) ein typisches Szenario gewählt, anhand dessen der prinzipielle Vorteil eines agentenbasierten Systems demonstriert wurde. In Anlehnung an dieses Vorgehen wird auch in der vorliegenden Arbeit auf der Basis von Architekturvergleichen eine qualitative Bewertung des agentenorientierten Ansatzes vorgenommen.

Die gesamte Arbeit wird als Grundlage für die Evaluierung durch das folgende Begründungsverfahren (Frank 2006, 220-320) geleitet, welches in Zusammenhang mit Design Science gesetzt wird und in Tabelle 1.9-2 zusammengefasst ist:

Die *Motivation* bzw. *Legitimation* für die Arbeit wird mit dem *Nutzen für die Praxis* begründet. Letzterer wird aus der Betrachtung des Status quo im Gesundheitswesen in Abschnitt 4 abgeleitet. Diese Anforderung geht auch mit dem Ansatz nach *Hevner et al.* (2004) konform: Darin werden die Anforderungen aus der Domäne abgeleitet.

Begründungskriterium	Detaillierung des Kriteriums	Bezug zu Design Science	Umsetzung in der vorliegenden Arbeit
Motivation/Legitimation	Nutzen für die Praxis	Begründung über die Bedürfnisse der Domäne	• Betrachtung des Status quo • Identifikation von Defiziten (Literaturanalyse) • Sachkundiges Argument
Spezifikation von Anforderungen	• Angemessenheit (im Hinblick auf Einsatzzweck) • Vorteilhaftigkeit (gegenüber existierenden Alternativen) • Nachhaltiger Bedarf in der Praxis	• Begründung über die Bedürfnisse der Domäne • Rückfluss der Ergebnisse in die Anwendungsdomäne bzw. die Wissensbasis	• Aufstellung des Anforderungskatalogs • Sachkundiges Argument
Beschreibung der Konstruktion	• Konzentration auf das Wesentliche • Begründung von Entwurfsentscheidungen	Bewertung der Ergebnisse	• Beschreibung wesentlicher Inhalte des Prototyps • Begründung von Entwurfs- und Implementierungsentscheidungen • Sachkundiges Argument (Argumentationslinie)
Überprüfung als wissenschaftliche Leistung	• Abstraktion, Generalisierungsanspruch • Originalität, Vergleich mit existierenden Alternativen • Begründung	Implizite Ableitung aus dem Bedarf bzw. dem Rückfluss in die Wissensbasis	• Sachkundiges Argument (Beleuchtung der Weiterentwicklungsmöglichkeit für zukünftige Informationssysteme) • Vergleich mit verwandten Arbeiten und Herausarbeitung des Alleinstellungsmerkmals (sachkundiges Argument)
Überprüfung anhand den Anforderungen	Mögliche Mängel kritisch diskutieren	Evaluierung als wesentliches Element	• Vergleich der Lösung mit dem Anforderungskatalog (sachkundiges Argument) • Diskussion von Mängeln • Erörterung möglicher Maßnahmen zur Beseitigung der Mängel
Aufzeigen von Erkenntnisfortschritt	• Delta aufzeigen • Delta bewerten	Implizit durch iterative Bewertung und Verfeinerung	• Bewertung des Erkenntnisfortschrittes • Sachkundiges Argument (Entwicklungspotenziale für Informationssysteme im Gesundheitswesen)

Tabelle 1.9-2: ***Zusammenfassung der Begründungszusammenhänge***
Quelle: Eigene Darstellung

Die *Spezifikation der Anforderungen* ist so zu gestalten, dass diese im *Hinblick auf den Einsatzzweck angemessen* sind. Die Spezifikation mündet in einen Anforderungskatalog für ein geeignetes Informationssystem, welcher in Abschnitt 4.12 beschrieben wird. Über den Vergleich mit existierenden Alternativen wird die *Vorteilhaftigkeit* der in der vorliegenden Arbeit beschriebenen Lösung herausgearbeitet. Die Anforderungen sind zudem so zu spezifizieren, dass der *Bedarf in der Praxis* von *nachhaltiger* Natur ist. Im Zusammenhang mit *Hevner et al.* (2004) sollen die Ergebnisse der Arbeit wieder zurück in das Anwendungsfeld bzw. die Wissensbasis fließen. Dazu kann der Vergleich mit aktuellen Informationssystemen im Ge-

sundheitswesen beitragen, aus dem letztendlich Entwicklungspotenziale für zukünftige Informationssysteme abgeleitet werden können.

Bei der *Beschreibung der Konstruktion* ist der *Fokus auf das Wesentliche* zu richten. Weiterhin sind bei *Entwurfsentscheidungen* entsprechende *Begründungen* anzugeben. Bei *Hevner et al.* (2004) ist eine damit vergleichbare Bewertung der Ergebnisse vorzunehmen, um diese intersubjektiv nachvollziehbar darzustellen.

Die *Überprüfung als wissenschaftliche Leistung* erfolgt anhand der Kriterien *Abstraktion* bzw. *Generalisierungsanspruch*, *Originalität* bzw. *Vergleich mit existierenden Alternativen* und einer *Begründung*. Bei *Hevner et al.* (2004) werden diese Aspekte nicht explizit genannt, können aber aus der Anforderung impliziert werden, dass der Bedarf abgeleitet werden muss bzw. in die Wissensbasis einfließen soll.

Bei der *Überprüfung* anhand der eingangs gestellten *Anforderungen* ist die Lösung entlang *möglicher Mängel kritisch zu diskutieren*. Auch bei *Hevner et al.* (2004) ist eine Evaluierung ein essenzieller Bestandteil der wissenschaftlichen Arbeit.

Um den *Erkenntnisfortschritt aufzuzeigen*, ist der in der vorliegenden Arbeit erreichte *Fortschritt aufzuzeigen* und zu *bewerten*. Diese Bewertung ist bei *Hevner et al.* (2004) durch die iterative Bewertung und Verfeinerung implizit gegeben.

Zusammengefasst werden die bisherigen Ausführungen zu den Forschungsfragen, Methoden und den angestrebten Ergebnissen im folgenden Abschnitt 1.10.

1.10 Forschungsmethoden zur Beantwortung der Forschungsfragen

Um die in Abschnitt 1.4 beschriebenen Forschungsfragen zu beantworten, werden die in Tabelle 1.10-1 dargestellten Methoden angewendet. Darin werden die jeweiligen Forschungsfragen mit den Forschungsmethoden und den daraus erarbeiteten Ergebnissen in Relation gesetzt.

Forschungsfrage	Forschungsmethode	Ergebnis
Ermittlung des Status quo und der Anforderungen im Gesundheitswesen	• Literaturanalyse • Sachkundiges Argument	Anforderungskatalog für ein Informationssystem
Erarbeitung eines Lösungskonzepts zur Überwindung ausgewählter Defizite	• Needs Driven Approach zur Orientierung in der zunächst unbekannten Domäne • ArBaCon-Methode zur architekturbasierten Konstruktion von Multiagentensystemen	Designvorschlag
Implementierung	Konstruktion von ausgewählten Elementen gemäß dem Prototypenbau und der ArBaCon-Methode	Prototypische Implementierung

Darstellung des Mehrwerts und Nutzens von aktiven medizinischen Dokumenten	Evaluierung mit geeigneten Methoden aus Design Science, ergänzt um weitere Methoden	• Bewertung der Ergebnisse • Vergleich mit dem Anforderungskatalog • Entwicklungsmöglichkeiten für künftige Informationssysteme im Gesundheitswesen

Tabelle 1.10-1: ***Forschungsfragen und Methoden zu ihrer Beantwortung***
Quelle: Eigene Darstellung

1.11 Annahmen und Abgrenzung der Arbeit

In diesem Abschnitt werden die für die vorliegende Arbeit angenommenen Annahmen beschrieben und eine Abgrenzung von in der Domäne eingesetzten Informationssystemen vorgenommen.

Die Ergebnisse dieser Arbeit und insbesondere die prototypische Implementierung grenzen sich von anderen, im Produktivbetrieb eingesetzten Informationssystemen dahingehend ab, dass erstere lediglich wissenschaftliche Ansätze darstellen, die die Grundlage für eine Weiterentwicklung bilden, die zum Einsatz der Ergebnisse in der Domäne qualifizieren. Die Implementierung dient somit der Demonstration der Tragfähigkeit der erarbeiteten Konzepte. Der Anspruch der Nutzungsevaluierung der prototypischen Implementierung wird mit der vorliegenden Arbeit nicht erhoben, da eine qualitative Bewertung des Nutzens der erarbeiteten Lösungskonzepte fokussiert wird. Deshalb wird die Implementierung nicht in Feld- oder Fallstudien evaluiert.

Weiterhin werden in der vorliegenden Arbeit nur solche Implementierungen erarbeitet, die lediglich die unmittelbar am Behandlungsprozess beteiligten Gesundheitsbetreuer und Patienten sowie die jeweiligen Informationsbedürfnisse berücksichtigen. Damit beschränkt sich die Arbeit auf die Bereitstellung von Informationssystemen für das medizinische und pflegerische Personal sowie für den Patienten. Die Bereiche der administrativen Verwaltung, Steuerung des Gesundheitswesens, Abrechnung sowie Kostenträger wird in der Implementierung nicht explizit betrachtet, obwohl mit der erarbeiteten Lösung auch Grundlagen für die in Abschnitt 1.1 beschriebene Steuerung des Gesundheitswesens gelegt werden (siehe dazu auch Abschnitt 8.3). Somit stellt diese Einschränkung keine Minderung der Bedeutung der Ergebnisse dar.

Weil die Anforderungen der TI für eine Umsetzung in der vorliegenden Arbeit zu umfangreich, dazu aktuell einer ständigen Weiterentwicklung unterworfen sind und die Spezifikationen für die Komponenten in unterschiedlichen Versionszuständen vorliegen, werden nur ausgewählte Details berücksichtigt, die wesentliche Voraussetzungen für die Etablierung einer integrierten Versorgung bilden. Auch diese Details erfüllen nicht den Anspruch der unmittelbaren und vollständigen Umsetzung der in den Spezifikationen für die TI beschriebenen Aspekte. Vielmehr soll exemplarisch demonstriert werden, wie die Möglichkeiten, die durch die TI gegeben sind, für den Mehrwertdienst einer virtuellen Patientenakte eingesetzt werden können, um ggf. Grundlagen für zukünftige Spezifikationen der Komponenten der TI zur Verfügung zu stellen.

Abschließend ist zu bemerken, dass in der vorliegenden Arbeit nicht der Anspruch erhoben werden kann, rechtliche sowie insbesondere datenschutzrechtliche Restriktionen vollständig zu berücksichtigen. Vielmehr wird in dieser Arbeit ein Konzept dargestellt, das in weiteren Arbeiten entsprechend den rechtlichen Vorgaben zu überprüfen und ggf. anzupassen ist. In diesem Zusammenhang wird für eine Betrachtung des Agentenbegriffes aus juristischer Perspektive auf die Ausführungen bei *Bergfelder/Nitschke/Sorge* (2005) sowie *Nitschke* (2006) verwiesen.

1.12 Aufbau der Arbeit

Der Aufbau der vorliegenden Arbeit (siehe Abbildung 1.12-1) orientiert sich an den Forschungsfragen aus Abschnitt 1.4 und wird im Folgenden beschrieben:

Ziel der Darstellung in Abschnitt 2 ist die Herausarbeitung von besonderen Eigenschaften von Software-Agenten gegenüber der herkömmlichen Software-Entwicklung. Dazu werden in Abschnitt 2.2 grundlegende Programmiertechniken in der Informatik beleuchtet. Diese Darstellung bildet die Grundlage für die Herausarbeitung der wesentlichen Charakteristika von Software-Agenten in Abschnitt 2.3. Dabei wird untersucht, unter welchen Voraussetzungen diesen welche Vorteilhaftigkeit gegenüber herkömmlichen Konzepten der Software-Entwicklung attribuiert werden kann.

Architekturen in der Software-Entwicklung werden in Abschnitt 3 beschrieben. Dazu werden zunächst allgemeine Architekturmuster in Abschnitt 3.2.3 dargestellt. Angesichts der Zielsetzung der vorliegenden Arbeit werden in Abschnitt 3.2.4 insbesondere Architekturen zur Integration von Informationssystemen beschrieben. In Abschnitt 3.3 werden agentenbasierte Architekturen betrachtet. Dabei werden diese Architekturen jeweils in Beziehung zu den in Abschnitt 3.2 beschriebenen Architekturen gesetzt. Damit wird das Ziel angestrebt, die Vorteilhaftigkeit der Realisierung von Software-Architekturen auf der Basis von Software-Agenten herauszuarbeiten. Die betrachteten Architekturen werden in Abschnitt 3.4 anhand eines Evaluierungskonzepts analysiert. Dabei ist qualitativ zu untersuchen, inwieweit Architekturen auf der Basis von Software-Agenten herkömmlichen Architekturen überlegen sein können.

In Abschnitt 4 werden wesentliche Eigenschaften der Domäne Gesundheitswesen beleuchtet. Dazu wird in Abschnitt 4.2 der Status quo im Gesundheitswesen dargestellt. Weiterhin werden in Abschnitt 4.3 Fehlerursachen in der Medizin abgeleitet. Das im Behandlungsprozess aktuell am häufigsten eingesetzte Medium zur Dokumentation sowie Kommunikation, Koordination und Kooperation der Papier basierten Patientenakte ist mit wesentlichen Nachteilen behaftet. Diese werden in Abschnitt 4.4 dargestellt. Aus den identifizierten Defiziten und Eigenschaften im Gesundheitswesen werden in den Abschnitten 4.5 und 4.6 die Potenziale digitaler Patientenakten bzw. der IT in der Medizin abgeleitet. In Abschnitt 4.7 werden die Defizite von Informationssystemen im Gesundheitswesen beleuchtet, die aus den Darstellungen zu den Potenzialen und dem bisherigen Status quo im Gesundheitswesen zu erschließen sind. Aus diesen Defiziten werden in den Abschnitten 4.8 mit 4.11 die allgemeinen Voraussetzungen für Informationssysteme im Gesundheitswesen, besondere Eigenschaften von institutionsübergreifender IT sowie die dafür erforderliche, einheitliche Kommunikationsinfrastruktur, die in der TI für die eGK gegeben ist, und Standards im Gesundheitswesen

beleuchtet. Aus diesen Anforderungen wird in Abschnitt 4.12 ein Anforderungskatalog für ein idealtypisches Informationssystem im Gesundheitswesen abgeleitet, in den entsprechend dem Aufbau der vorliegenden Arbeit auch der Status quo des Gesundheitswesens einfließt. Mit diesem Anforderungskatalog ist Forschungsfrage 1 aus Abschnitt 1.4 zu beantworten.

Auf der Grundlage der Ausführungen in den Abschnitten 2 mit 4 ist in Abschnitt 5 zu begründen, dass sich ein Multiagentensystem (MAS) vorteilhaft für die Umsetzung von Anforderungen im Gesundheitswesen eignet. Dazu wird in Abschnitt 5.2 die Modellierung des Gesundheitswesens als CAS (Complex Adaptive System) dargestellt. In Abschnitt 5.3 wird gezeigt, dass sich ein CAS geeignet durch ein MAS auf Ebene der Software-Konstrukte abbilden lässt. Somit lassen sich die Eigenschaften der Domäne Gesundheitswesen geeignet auf ein MAS übertragen, womit die Grundlage für die Konstruktion eines geeigneten Informationssystems für diese Domäne gegeben ist. In Abschnitt 5.4 wird beschrieben, wie der Anforderungskatalog aus Abschnitt 4.12 agentenbasiert realisiert werden kann. Mit diesen Ausführungen sind nun die Grundlagen für einen geeigneten Anwendungsfall gegeben, an dem demonstriert werden soll, welche Eigenschaften ein agentenbasiertes System besitzen muss, um den Anforderungskatalog aus Abschnitt 4.12 zu erfüllen.

In Abschnitt 6 wird beschrieben, welches Lösungskonzept die identifizierten Defizite beseitigen sowie welche Eigenschaften eine agentenbasierte Implementierung für ein zukünftiges Informationssystem im Gesundheitswesen besitzen kann. Dazu werden in Abschnitt 6.2 zunächst Domänen bezogene Rahmenbedingungen des gewählten Anwendungsfalles beschrieben. Ergänzt werden diese Ausführungen durch die Darstellung von technischen Eigenschaften einer möglichen Implementierungsplattform in Abschnitt 6.3. Auf der Grundlage der agentenbasierten Realisierung des Anforderungskatalogs in Abschnitt 5.4 sowie der Beschreibung von Anwendungsfall spezifischen und technischen Aspekten in den Abschnitten 6.2 und 6.3 wird in Abschnitt 6.3.6 der Anforderungskatalog konkretisiert. Aus den Abschnitten 6.2 mit 6.3.6 lässt sich in Abschnitt 6.5 die Gesamtarchitektur des Informationssystems ableiten. Ein wesentliches Element ist dabei das Konzept der aktiven medizinischen Dokumente (AMD). Ausgangspunkt für die Darstellungen der vorliegenden Arbeit zu aktiven medizinischen Dokumenten sind die bei *Krcmar/Horn* (1999, 6-7; 2001a, 18) initiierten und bei *Krcmar/Horn* (2001a, 7 und 12-13; 2001b, 4, 12-13 und 15-16; 2003, 2, 5, 11 und 43), *Reinke et al.* (2002; 2003) sowie *Wilczek* (2007) erarbeiteten Ergebnisse. Diese Resultate werden bei *Schweiger/Bastian/Krcmar* (2005), *Schweiger/Krcmar* (2006, 308-309) und in der vorliegenden Arbeit weiterentwickelt. In den Abschnitten 6.6 und 6.7 werden in einen Ausschnitt aus einem gewählten Szenario der OP-Vorbereitung die vorgeschlagene Gesamtarchitektur sowie das Design der gesamten agentenbasierten Applikation eingebettet. Mit der Beschreibung des Designs in Abschnitt 6.7 ist die Forschungsfrage 2 aus Abschnitt 1.4 zu beantworten. Die Ausführungen zum Design fließen in die Darstellung der Implementierung in Abschnitt 6.8, durch die die Forschungsfrage 3 aus Abschnitt 1.4 zu beantworten ist.

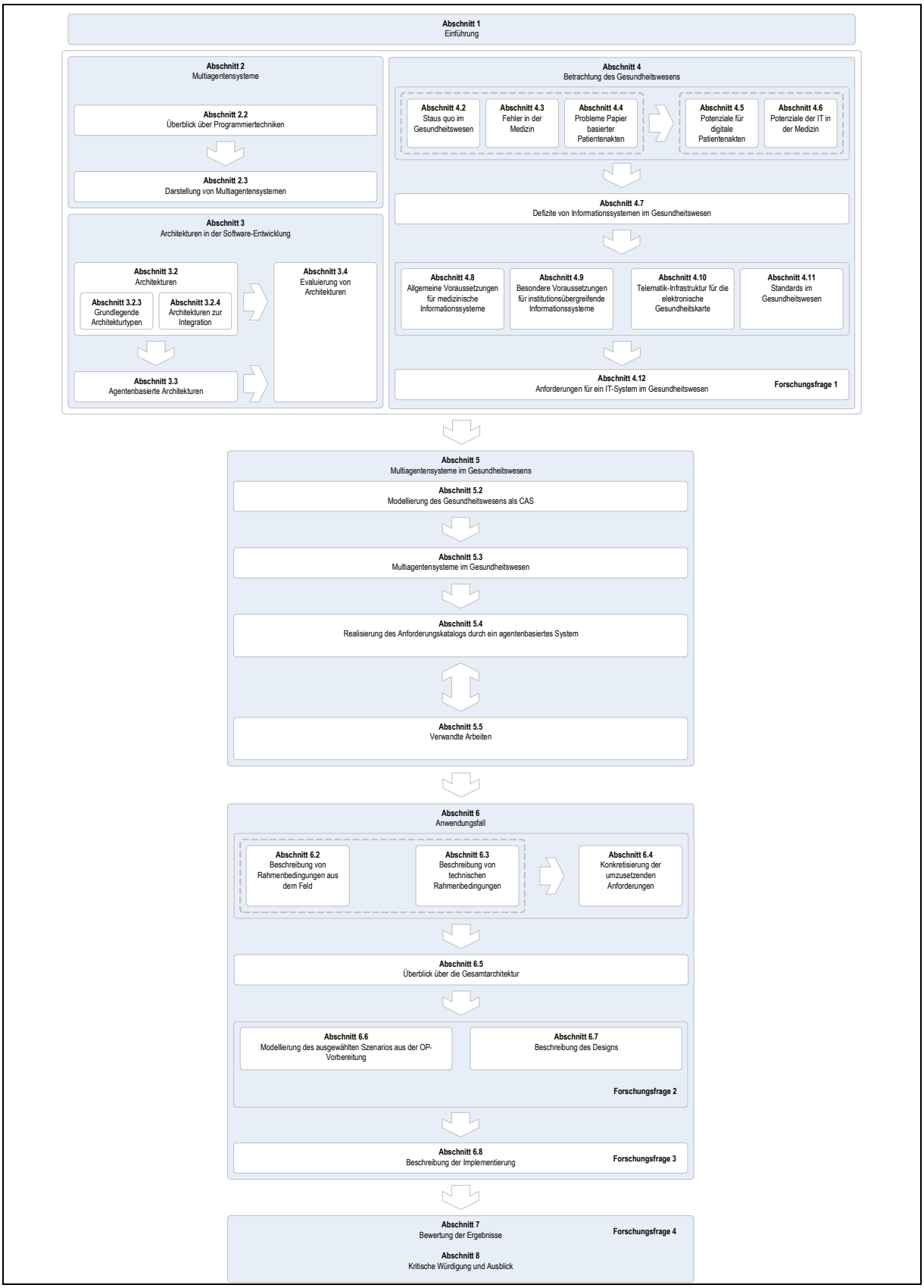

Abbildung 1.12-1: Überblick über den Aufbau der Arbeit
Quelle: Eigene Darstellung

In Abschnitt 7 wird die Bewertung der Ergebnisse aus technischer, domänenspezifischer und Anwenderperspektive vorgenommen. Weiterhin wird insbesondere auch die wissenschaftliche Leistung der vorliegenden Arbeit herausgearbeitet. Ausgehend von der Beschreibung der Entwicklungsperspektiven für zukünftige Informationssysteme im Gesundheitswesen wird außerdem der Mehrwert durch das in der vorliegenden Arbeit auf der Basis der Arbeiten von *Krcmar/Horn* (1999, 6-7; 2001a, 7, 12-13 und 18; 2001b, 4, 12-13 und 15-16; 2003, 2, 5, 11 und 43), *Reinke et al.* (2002; 2003) und *Wilczek* (2007) weiterentwickelte AMD-Konzept herausgearbeitet. Mit diesen Ausführungen ist Forschungsfrage 4 aus Abschnitt 1.4 zu beantworten.

Die Arbeit wird mit einer Zusammenfassung, einer Beschreibung des Erkenntnisfortschrittes sowie einem Ausblick für zukünftige Arbeiten in Abschnitt 8 abgeschlossen.

2 Multiagentensysteme

2.1 Überblick

Ziel des Abschnittes 2 ist die Herausarbeitung der wesentlichen Unterschiede zwischen der herkömmlichen Software-Entwicklung und derjenigen auf der Basis von Software-Agenten (siehe Abbildung 2.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1). Dazu wird zunächst in Abschnitt 2.2 ein Überblick über Programmiertechniken gegeben, um die zunehmende Abstraktion in der Informatik zu demonstrieren. In Abschnitt 2.3 werden die Eigenschaften von Software-Agenten beschrieben, um zu zeigen, unter welchen Bedingungen diese Technologie vorteilhaft eingesetzt werden kann.

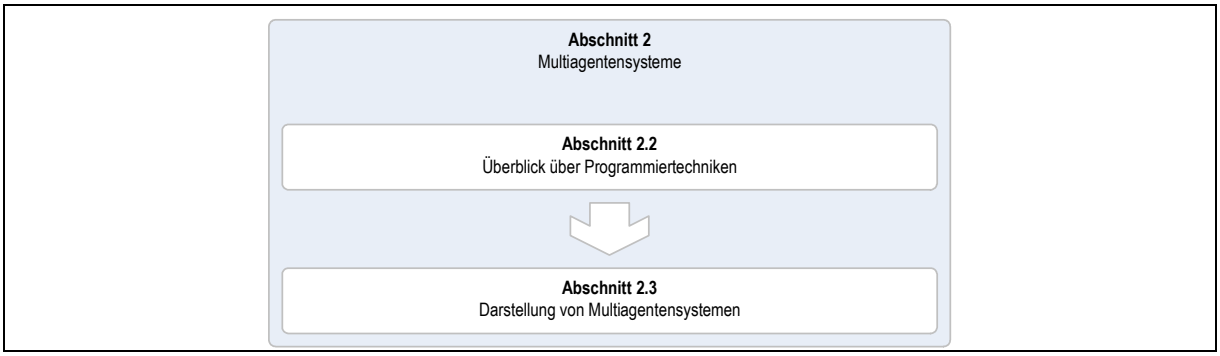

Abbildung 2.1-1: ***Überblick über Abschnitt 2***
Quelle: Eigene Darstellung

2.2 Überblick über Programmiertechniken

2.2.1 Abstraktion in der Informatik

Die Komplexität von Rechensystemen weist eine stetige Zunahme auf (Broy/Rumpe 2007, 3), welche durch die beiden folgenden Faktoren beeinflusst wird. Nach *Moore* (1976) nimmt die Leistung von Prozessoren exponentiell über die Zeit betrachtet zu. Die von Moore ursprünglich getroffene Aussage wurde aus Erfahrungswerten abgeleitet, um die bisherige Hardware-Entwicklung aufzuzeigen. Heute strebt die Industrie jedoch selbst die weitere Erfüllung von Moores Gesetz an. Neben der zunehmenden Leistungsfähigkeit von Prozessoren ist eine signifikante Zunahme der Übertragungskapazitäten von Kommunikationsnetzen zu beobachten. Die Abschätzung durch das *Bundesamt für Sicherheit in der Informationstechnik* (2003, 156-157) lässt ein exponentielles Wachstum sowohl im leitergebundenen wie im drahtlosen Bereich erwarten. Die Entwicklung von zunehmend leistungsfähiger Hardware und ihre Vernetzung bilden die Voraussetzung für die Konstruktion von Software mit umfangreicher Funktionalität und damit einhergehender, wachsender Komplexität. Neben der Leistungsfähigkeit der Hardware wird die Komplexität von Software-Systemen auch von der Notwendigkeit getrieben, komplexe technische Systeme mir ihren Funktionalitäten zu beherrschen. Solche Systeme sind ohne geeignete Software teilweise nicht mehr zu realisieren.

Der Komplexität von Software-Systemen wurde hinsichtlich ihrer Entwicklung mit unterschiedlichen Ansätzen begegnet: Einerseits durchlief die Software-Entwicklung die Phasen

von der maschinennahen Programmierung bis zum etablierten objektorientierten Paradigma. Dabei wurde eine immer höhere Abstraktion von maschinenspezifischen Merkmalen erreicht und damit die vereinfachte sowie Plattform unabhängige Konstruktion von Software unterstützt. Mit dem Ansatz Model Driven Architecture® der Object Management Group®[6], der bereits seit geraumer Zeit untersucht wird (Loos/Scheer 1995), wird eine weitere Abstraktionsebene hinzugefügt. Dieser Ansatz abstrahiert von der eingesetzten Implementierungstechnologie und ermöglicht eine automatisierte Transformation von Geschäftsmodellen in Konstrukte der Software-Technik sowie ausführbare Software-Systeme (Kempa/Mann 2005, 298). Andererseits konnte mit der durch die unterschiedlichen Programmierparadigmen erzielten Modularisierung die Komplexität von Software-Systemen reduziert bzw. beherrschbar gestaltet werden (Broy/Rumpe 2007, 4).

Obwohl mit der zunehmenden Abstraktion der Programmiertechniken eine gleichzeitige Unabhängigkeit von der zugrunde liegenden Maschinenarchitektur verbunden ist, lässt sich keine generelle Empfehlung für oder wider die Selektion einer bestimmten Technik geben. Dies ist damit begründet, dass nicht nur das Abstraktionsniveau ein Auswahlkriterium ist, sondern oftmals auch externe Treiber wesentlichen Einfluss besitzen. Darunter fallen z.B. Vorgaben eines Auftraggebers, die gegebene Aufgabenspezifikation, die bereits vorhandene Informationssystemlandschaft oder Hardware-Plattformen. Eine Entscheidung für oder gegen eine bestimmte Vorgehensweise ist deshalb jeweils individuell zu treffen. Im Folgenden werden als Entscheidungsgrundlage wesentliche Programmiertechniken jeweils zusammen mit ihren Vor- und Nachteilen beschrieben.

2.2.2 Maschinennahe Programmierung

Die maschinennahe Programmierung (siehe dazu auch Broy 1998, 403-455) orientiert sich mit ihren Konstrukten an der jeweils vorliegenden Maschinenarchitektur. Letztere gibt den möglichen Befehlsvorrat für Programme vor. Durch diesen werden die Struktur, Form und Bedeutung der Maschinenbefehle bestimmt (Broy 1998, 403).

Die mit maschinennahen Programmen formulierten Berechnungsvorschriften sind an das Verarbeitungsmodell der eingesetzten Maschine angelehnt und werden sequentiell notiert und abgearbeitet. Die Operanden der Befehle werden als Konstanten, in Registern oder mit einer Adresse spezifiziert, die auf einen Speicherblock im Arbeitsspeicher zeigt (eine ausführliche Darstellung zur Operandenspezifikation findet sich bei Broy 1998, 423-431). Das Aufbrechen der rein sequentiellen Abarbeitung der Programme, die auch im Speicher sequentiell abgelegt werden, kann durch eine Strukturierung von Programmen mit (bedingten) Sprüngen erfolgen (Broy 1998, 403). Durch Erweiterungen der Maschinensprache entstehen Assemblersprachen (Broy 1998, 415). Dabei wird insbesondere die weitere Strukturierungsmöglichkeit der Unterprogrammaufrufe unterstützt (Broy 1998, 418-423).

[6] *Object Management Group* (2000; 2001a; 2001b; 2003), *Kleppe/Warmer/Bast* (2003).

Vorteile	Nachteile
• Äußerst effiziente Abarbeitung von Berechnungsvorschriften, insbesondere bei genauer Kenntnis der Maschinenarchitektur • Relevant für die Implementierung von Software für eingebettete und Echtzeitsysteme • Effiziente Ressourcenauslastung • Ermöglicht Implementierung bei Abwesenheit eines geeigneten Übersetzers für Hochsprachen • Direkter Zugriff auf die Hardware, der durch eine Hochsprache nicht realisierbar wäre	• Schwierig zu erstellen (Broy 1998, 403) • Erfordert das Erstellen von umfangreichen Programmen • Schwierig lesbar (Broy 1998, 403) • Hohe Fehleranfälligkeit • Beschränkte Übertragbarkeit auf andere Rechnerarchitekturen • Geringe Wiederverwendung der Programmteile • Strukturierungsmöglichkeit nur durch (bedingte) Sprünge und Unterprogrammaufrufe

Tabelle 2.2-1: ***Vor- und Nachteile der maschinennahen bzw. Assemblerprogrammierung***
Quelle: Eigene Darstellung

Bestimmte Anforderungen an ein zu konstruierendes Software-System können den Einsatz der maschinennahen oder Assemblerprogrammierung bedingen. Dennoch sind mit der Implementierung eines Programms mit dieser Programmiertechnik Nachteile assoziiert, die es anraten, diese Programmiertechnik zugunsten einer höheren Programmiersprache zu vermeiden (Broy 1998, 403). Die Vor- und Nachteile der maschinennahen bzw. Assemblerprogrammierung sind in Tabelle 2.2-1 dargestellt.

2.2.3 Funktionale Programmierung

Ein funktionales (applikatives) Programm wird nach *Broy* (1998, 89-90) als eine Menge von Funktionsdefinitionen und ihrer Applikation in Termen verstanden. Die Aufrufe von Funktionen liefern als Ergebnis einen Wert zurück. Bei Termen werden primitive und nicht primitive Terme bzw. Ausdrücke differenziert (Broy 1998, 90). Erstere enthalten nur Operationssymbole aus der gegebenen Rechenstruktur, letztere umfassen auch die Sprachelemente der funktionalen Programmiersprache (Broy 1998, 90-91). Die Programmelemente der funktionalen Programmierung (siehe dazu auch Broy 1998, 89-110) sind in Tabelle 2.2-2 dargestellt. Eine Ausführung eines applikativen Programms entspricht somit der Auswertung eines Ausdrucks (Broy 1998, 89).

Funktionales Programmelement	Beschreibung
• Funktionsdeklaration • Funktionsapplikation	• Definition der Berechnungsvorschrift einer Funktion • Aufruf einer Funktion
Elementdeklaration	Vereinbarung eines Elements
Konstante	Semantische Interpretation ist invariant (Broy 1998, 95)
Identifikatoren	Bezeichnung von Zwischenergebnissen, Platzhalter in Ausdrücken, Bezeichnung von Funktionen (Broy 1998, 94), Referenzierung von Werten (jedoch keine Veränderung der Werte)
Bedingte Ausdrücke	Steuerung des Kontrollflusses entsprechend einer Fallunterscheidung (Broy 1998, 95)

Tabelle 2.2-2: ***Elemente der applikativen Programmierung***
Quelle: Eigene Darstellung, zusammengefasst aus den Ausführungen bei *Broy* (1998, 89-110)

Ein wesentliches Element der applikativen Funktionsdeklaration ist die Rekursion. In rekursiven Funktionsdeklarationen wird eine Referenz auf die Deklarationsvorschrift selbst angegeben. Mit dieser mächtigen Darstellungsform von Rechenvorschriften lassen sich unbeschränkt lange Berechnungen formulieren (Broy 1998, 110). Entsprechend der Struktur der Rekursion wird eine Reihe unterschiedlicher Rekursionsformen unterschieden. Weil eine ausführliche Darstellung dieser Arten und von Techniken der semantischen Deutung von rekursiven Berechnungsvorschriften für die vorliegende Arbeit nicht relevant ist, wird dazu auf *Broy* (1998, 110-131) verwiesen.

Mit der applikativen Programmierung wird von der zugrunde liegenden Hardware-Architektur abstrahiert und damit eine erste Stufe der Abstraktion eingeführt. Somit können mit der funktionalen Programmierung die in Tabelle 2.2-1 genannten Nachteile der maschinennahen bzw. Assemblerprogrammierung beseitigt werden.

Die applikative Programmierung ist an die mathematische Darstellung von Funktionen angelehnt. Deshalb eignet sich diese Programmiertechnik insbesondere für die mathematische Beweisführung der Korrektheit sowie Terminierung und damit der Verifikation von Programmen. Weiterhin ist festzustellen, dass die in der funktionalen Programmiertechnik vorherrschenden Konzepte auch in anderen Techniken Verwendung finden und eine klare Darstellung der Algorithmen erfolgen kann (Broy 1998, 89).

Nachteilig für applikative Programme kann festgestellt werden, dass die ihr zugrunde liegende Denkweise für den Implementierer im Vergleich zur prozeduralen Programmierung (siehe Abschnitt 2.2.4) ungewohnt ist (Broy 1998, 155). Weiterhin ist die Strukturierung in funktionalen Programmen mit dem Strukturierungselement Funktion eingeschränkt. Damit eignet sich diese Programmiertechnik weniger gut zur Reduktion der Komplexität von sehr umfangreichen Programmen.

2.2.4 Prozedurale Programmierung

Bei der Abarbeitung eines Programms in Assemblersprache werden in einem Befehlszyklus die Inhalte der Register als Zwischenspeicher, Prozessorstatusregister, Hilfsregister und Speicherzellen verändert (Broy 1998, 405). Umgekehrt steuert der Zustand der Register und Speicherzellen den weiteren Verlauf der Berechung (Broy 1998, 155), z.B. als Grundlage für eine bedingte Anweisung. Somit kann festgestellt werden, dass der Zustand der Maschine den Fortgang der Berechnungen steuert und jeder Befehlszyklus wiederum die Inhalte der Speicherzellen und Register beeinflusst (siehe dazu auch Broy 1998, 155). Eine möglichst effiziente Unterstützung dieses Maschinenmodells auf der Ebene der Programmiertechnik wird durch die prozedurale Programmierung ermöglicht (Broy 1998, 155). Dabei ist charakteristisch, dass jede Anweisung Teile des Speicher- und Ablaufzustandes verändert (Broy 1998, 155).

Unter einem prozeduralen Programm wird eine Folge von Anweisungen verstanden (Broy 1998, 155). Der Zustand des Programms wird durch die aktuelle Belegung seiner Programmvariablen definiert (Broy 1998, 157). Der Endzustand für nicht terminierende Programme wird mit $\perp$ („Bottom“, Broy 1998, 46) bezeichnet (Broy 1998, 157-158). Die Elemente der prozeduralen Programmierung sind in Tabelle 2.2-3 dargestellt.

Die Besonderheiten der prozeduralen Programmierung, Seiteneffekte durch globale Variablen und Überlagerung von Identifikatoren (Broy 1998, 174), erfordern zusätzlich zu den von applikativen Programmiersprachen übernommenen Techniken der Induktion (Broy 1998, 146-149), Anwendung von Fixpunkteigenschaften (Broy 1998, 149-150) und Terminierungsfunktionen (Broy 1998, 150-153) die Einführung spezieller Vorgehensweisen zur Verifikati-

on. Die Methode der Zusicherung (Hoare/Wirth 1973; Broy 1998, 174-175) orientiert sich dabei an folgendem Vorgehen:

Prozedurales Programmelement	**Beschreibung**
Programmvariable	Speicherung von (variablen) Zwischenergebnissen, Referenzierung von Speicheradressen
Variablendeklaration und Zuweisung	Vereinbarung von Variablen und Belegung mit einem Wert
nop	Leere Anweisung (Identitätsabbildung hinsichtlich des Zustandes der Maschine)
abort	Nicht terminierende Anweisung zur Abbildung eines beliebigen Zustandes auf den nicht definierten Zustand ⊥ (Broy 1998, 159)
Sequentielle Komposition	Sequentielle Aneinanderreihung von prozeduralen Programmelementen
Zusammengesetzte Anweisung	Sequentielle Aneinanderreihung von Anweisungen
Bedingte Anweisung	Steuerung des Kontrollflusses entsprechend den Ergebnissen einer Fallunterscheidung (Broy 1998, 95)
Wiederholungsanweisungen	• Abweisende Wiederholung: Wiederholte Ausführung des Schleifenrumpfes, solange die Schleifenbedingung den Wahrheitswert **true** besitzt (Broy 1998, 163) • Nicht abweisende Wiederholung: Erst nach der erstmaligen Ausführung des Schleifenrumpfes erfolgt eine Überprüfung, ob die Schleifenbedingung erfüllt ist; besitzt die Abbruchbedingung den Wahrheitswert **true**, wird die Wiederholung abgebrochen (Broy 1998, 165). • Gezählte Wiederholung: Angabe der Wiederholungsschritte einer Schleife (Broy 1998, 165)
Prozedurdeklaration	Sammlung von Anweisungen, die mehrmals ausgeführt werden sollen, wobei eine Prozedur keinen expliziten Rückgabewert besitzt (Broy 1998, 168). Die Manipulation der Variablen erfolgt über die im Prozeduraufruf übergebenen Parameter oder über globale Variablen.
Prozeduraufruf	Aufruf einer Prozedur

Tabelle 2.2-3: ***Elemente der prozeduralen Programmierung***
Quelle: Eigene Darstellung, zusammengefasst aus den Ausführungen bei *Broy* (1998, 155-172)

- Annotierung von Programmen mit Prädikaten als Kommentare, welche Variablennamen aus dem Programm als freie Identifikatoren besitzen
- Aus einer Zusicherung als Vorbedingung und der jeweiligen Anweisung wird eine Zusicherung als Nachbedingung abgeleitet.
- Ein Kommentar kann mit dem Wahrheitswert **true** belegt werden, falls der Kommentar für jeden an dieser Stelle möglichen Zustand korrekt ist.

- Falls jeder Kommentar mit dem Wahrheitswert **true** belegt werden kann, ist die Korrektheit des Programms verifiziert.

Als Unterstützung für den Prozess der Verifikation werden spezielle Ableitungsregeln angegeben, mit deren Hilfe der mathematische Beweis im Rahmen der Zusicherungsmethode durchgeführt werden kann (siehe dazu die ausführlichen Darstellungen bei Broy 1998, 176-178).

Mit der applikativen und prozeduralen Programmierung wird die Programmierung unabhängig von der zugrunde liegenden Hardware-Struktur. Das Abstraktionsniveau wird demzufolge angehoben. Zusammenfassend kann im Vergleich der applikativen und prozeduralen Programmierung festgestellt werden, dass letztere als im Wesentlichen applikative Programmierung mit der Einschränkung der Nutzung nur der repetitiven Rekursion[7] verstanden werden kann (Broy 1998, 155). Weiterhin ist anzumerken, dass der funktionalen im Vergleich zur applikativen Programmierung die Zuweisung sowie die Wiederholungsanweisung fehlen.

Weil die „Verwendung von Anweisungen zur Kontrolle und Steuerung von Abläufen" (Broy 1998, 155) bekannt und geläufig ist, erscheint die prozedurale im Vergleich zur applikativen Programmierung intuitiver (Broy 1998, 155). Als vorteilhaft können weiterhin die besondere Effizienz der zuweisungsorientierten Programmierung wegen der Ähnlichkeit hinsichtlich des Zustandsmodells der zugrunde liegenden Maschine sowie die Modularisierung durch Prozeduren genannt werden. Nachteilig kann sich der Zugriff auf globale Variable auswirken, weil damit ggf. ungewollte Seiteneffekte entstehen.

2.2.5 Objektorientierte Programmierung

Im folgenden Abschnitt werden elementare Aspekte der Objektorientierung beschrieben. Außerdem werden die weiteren Eigenschaften fokussiert: Anreicherung mit Schlüsselwörtern, Fehlerbehandlung und Kommunikation zwischen Objekten.

Die objektorientierte Programmierung[8] stellt eine Weiterentwicklung der prozeduralen Programmierung hinsichtlich methodischer Grundsätze dar (Broy 1998, 247). Damit wird einerseits die Simplifizierung der Abbildung von Geschäftsobjekten auf Konstrukte der Software-Technik unterstützt. Andererseits wird der Entwicklungsprozess besonders berücksichtigt. Objektorientierte Sprachen besitzen insbesondere Elemente, die die folgenden Aspekte unterstützen (Broy 1998, 247) und im Weiteren beschrieben werden:

- Klassen zur Kapselung von Daten und Prozeduren
- Objekte als Instanzbildungen von Klassen (dynamische Generierung von Objekten)
- Geheimnisprinzip und Datenabstraktion
- Persistenz
- Vererbung und Polymorphie

Ein objektorientiertes Programm besteht aus einer Menge von Klassen (Broy 1998, 247). Diese Klassen stellen Baupläne für zur Laufzeit generierte Objekte dar. Objekte stellen somit spezielle Ausprägungen der Klassen mit der jeweils aktuellen Belegung ihrer Attribute mit

[7] Eine Rekursion ist repetitiv, wenn in einer linear rekursiven Funktionsvereinbarung „in allen rekursiven Zweigen einer Fallunterscheidung der rekursive Aufruf als äußerste (letzte) Aktion" (Broy 1998, 126) auftritt. Lineare Rekursionen besitzen in jedem Zweig einer Fallunterscheidung höchstens einen rekursiven Aufruf der Funktion (1998, 127).

[8] *Claussen* (1993), *Mössenböck* (1994), *Budd* (1996), *Meyer* (2000).

Werten dar (Broy 1998, 255). Sie enthalten als Programmeinheit die folgenden Elemente (Broy 1998, 247-248):

- Funktionen, Prozeduren (Methoden)
- Konstanten, Programmvariablen (Attribute)

In einer Klasse werden damit Methoden und Attribute gekapselt. Mit dem Klassennamen wird gleichzeitig ein Datentyp eingeführt (Broy 1998, 250). Objektidentifikatoren verweisen als ein Zeiger auf die aus einer Klasse generierten Objekte (Broy 1998, 251). Zur Realisierung des Geheimnisprinzips können Attribute nur über Methodenaufrufe manipuliert werden. Die Methoden stellen dabei die syntaktische Schnittstelle oder Nutzungssicht einer Klasse dar (Broy 1998, 247 und 252). Im Gegensatz dazu steht die Realisierungssicht einer Klasse (Broy 1998, 247), die die interne Struktur der Klasse repräsentiert.

Weil die Attribute hinsichtlich ihrer Lebensdauer über einen Methodenaufruf hinaus fortbestehen, wird von Persistenz der Objekte gesprochen, insbesondere dann, wenn Objekte über die Laufzeit eines Programms hinaus existieren (Broy 1998, 254-255).

Die Relation zwischen zwei Klassen, die in der Beziehung Oberklasse bzw. Verallgemeinerung und Unterklasse bzw. Spezialisierung stehen, wird mit Vererbung bezeichnet. Dabei werden alle vererbbaren Attribute und Methoden einer Oberklasse an die Unterklasse weitergereicht. Somit können Klassen, die allgemeine Eigenschaften besitzen, von mehreren Unterklassen wieder verwendet und dadurch spezialisiert werden. Weiterhin wird mit der Vererbung eine verbesserte Strukturierung eines Programms erreicht (Broy 1998, 255). Aufgebaute Vererbungshierarchien sind transitiv (Broy 1998, 256). Damit kann festgestellt werden, dass in einem objektorientierten System alle Objekte einer gemeinsamen Wurzel zugeordnet werden. Folglich kann ein objektorientiertes Software-System als ein Beziehungsgeflecht zwischen den existierenden Objekten betrachtet werden. Bei diesen Vererbungsbeziehungen wird zwischen singulärer und multipler Vererbung unterschieden (Broy 1998, 256). Bei einfacher Vererbung steht eine Unterklasse mit höchstens einer Oberklasse in Relation. Multiple Vererbung ist durch die Beziehung gekennzeichnet, die aus einer Unterklasse und mindestens zwei vererbenden Oberklassen entsteht. Die vererbten Methoden können in einer Unterklasse überladen (Broy 1998, 258), d.h. neu definiert werden. In Abhängigkeit vom Typ eines Objekts, auf dem eine Methode aufgerufen wird, wird die jeweils zugehörige Methode der Ober- oder Unterklasse aufgerufen. Es wird damit erst zur Laufzeit entschieden, welche Methode aufgerufen wird. Dies wird als dynamische Bindung bzw. Polymorphie bezeichnet (Broy 1998, 259).

Eine Reihe von Schlüsselwörtern erweitert die Möglichkeiten der Objektorientierung, indem die Instanzbildung, Sichtbarkeit und Vererbungsvorschriften spezifiziert werden. Mit der Einführung dieser Aspekte kann eine verbesserte Strukturierung der Programme erreicht werden (Broy 1998, 260). Abstrakte Klassen können nicht deklarierte Methodenrümpfe enthalten und können deshalb nicht als Grundlage einer Instanzbildung dienen (Broy 1998, 259). In der Vererbungshierarchie dieser Klassen müssen die abstrakten Methodenrümpfe definiert werden, bevor daraus Objekte erzeugt werden können. Finale Methoden können in einer Unterklasse nicht überschrieben werden (Broy 1998, 259). Weitere Schlüsselwörter eröffnen Mög-

lichkeiten der Sichtbarkeitsbestimmung von Methoden und Attributen über ihre Kennzeichnung z.B. als öffentlich oder privat (Broy 1998, 260). Öffentliche Methoden und Attribute sind außerhalb von einer Klasse sichtbar. Im Gegensatz dazu kann auf private Methoden und Attribute nur innerhalb der jeweiligen Klasse zugegriffen werden. Zusätzlich kann durch Schlüsselwörter festgelegt werden, welche Attribute und Methoden in die Unterklasse vererbt werden (Broy 1998, 260).

Die Objektorientierung führt besondere Elemente der Fehlerbehandlung ein. Treten bei der Ausführung einer Methode Fehler auf, können diese gesondert über so genannte Ausnahmen behandelt werden (Broy 1998, 261). Damit wird eine klare Trennung zwischen Verarbeitungsvorschriften und Fehlerbehandlung erreicht.

Objekte kommunizieren über den Versand von Nachrichten, die im Methodenaufruf in dem Zielobjekt manifest werden. Sobald die Berechungsvorschrift in der aufgerufenen Methode abgearbeitet ist, wird ggf. ein Ergebnis an das aufrufende Objekt zurückgeliefert. Objekte sind also in ihrem Verhalten per se statisch, weil sie nur auf eingehende Nachrichten reagieren. Ein aktives Verhalten von Methoden kann der Objektorientierung nicht attribuiert werden.

Insgesamt kann festgestellt werden, dass mit der Objektorientierung eine weitere Stufe der Abstraktion eingeführt wird. Dazu können Geschäftsobjekte intuitiv in Strukturen der Software-Technik abgebildet werden. Zudem zielt die Objektorientierung bis zu einem bestimmten Grad auf eine verbesserte Wiederverwendung und Strukturierung ab (Broy 1998, 261) und kann so die Beherrschbarkeit der Komplexität von Software-Systemen erhöhen. Die Wiederverwendung wird durch die stark ausgeprägte Verkapselung von Daten und Methoden in Klassen unterstützt. Damit wird die Implementierung vom Nutzer der Klasse verborgen und kann sogar geändert werden, ohne Auswirkungen auf die Nutzung der Schnittstelle zu haben.

Obwohl ein Ziel der Objektorientierung die bessere Wiederverwendung ist, kann dennoch beobachtet werden, dass die Abstraktion und Granularität, die für die Nutzer der Funktionalität eines Objekts zur Verfügung gestellt werden, zu fein granular sind, um effiziente Wiederverwendung oder auch Verteilung zu unterstützen (Krafzig/Banke/Slama 2006, 18). Die Objektorientierung kann deshalb als ein ausreichendes Paradigma lediglich für die Implementierung von umfangreichen isolierten und monolithischen Anwendungen betrachtet werden (Krafzig/Banke/Slama 2006, 18). Die Service-Orientierung (siehe dazu die Ausführungen in Abschnitt 3.2.4.2) kann jedoch einen Beitrag leisten, um eine geeignete Granularität zu realisieren.

Mit den Darstellungen über Programmiertechniken ist die Grundlage für die Beschreibung von agentenbasierten Systemen im folgenden Abschnitt gegeben, die in einen Vergleich mit der Objektorientierung mündet, die als ein etablierter Ansatz der Software-Entwicklung gilt.

2.3 Darstellung von Multiagentensystemen

Software-Agenten können als eine Weiterentwicklung von Objekten aus der Objektorientierung verstanden werden. Bei Agenten werden insbesondere die aktiven Elemente im Gegensatz zu Software-Objekten hervorgehoben. Für eine einheitliche Definition eines Software-Agenten existiert bisher keine etablierte Begriffsbildung. Dies ist bei näherer Betrachtung auch nachvollziehbar, befassen sich doch eine Reihe unterschiedlicher Disziplinen mit Agentenkonzepten (Pokahr/Braubach/Lamersdorf 2005a, 301). Demzufolge werden in der Literatur unterschiedliche Definitionsansätze und damit differenzierte Konzepte identifiziert, die im Folgenden überblicksartig dargestellt werden. Beschrieben werden die Ansätze eines minimalen Agentenbegriffs, von rationalen Agenten, Agenten mit Intelligenzaspekten sowie mit humanen Eigenschaften. Diese Beschreibungen stellen die Grundlage für die Auswahl einer geeigneten Terminologie für Agenten in Abschnitt 5.3 dar, die für die vorliegende Ar-

beit adäquat ist. Weiterhin werden relevante Grundlagen von agentenbasierten Software-Systemen beschrieben sowie eine Gegenüberstellung zwischen Agenten und Objekten detailliert, um die Vorteilhaftigkeit der Agententechnologie herauszuarbeiten.

2.3.1 Klassifikation von Software-Agenten

2.3.1.1 Minimaler Agentenbegriff

Der minimale Agentenbegriff ist derjenige, welcher in der Literatur sehr häufig zur Definition von Software-Agenten herangezogen wird und wenig umfangreiche Anforderungen an die Eigenschaften eines Software-Agenten beschreibt. Demnach können Software-Agenten als Computer-Systeme mit den folgenden Eigenschaften verstanden werden (Wooldridge/Jennings 1995, 118):

- *Autonomie*: Bei nicht agentenorientierter Software werden die Reaktionen eines Systems vom Programmierer im Voraus festgelegt. Im Gegensatz dazu entscheiden Software-Agenten autonom, welche Aktionen durchgeführt werden. Agenten handeln außerdem ohne die direkte Intervention von Menschen und besitzen die Kontrolle über ihre eigenen Aktionen und ihren internen Zustand. Damit sind Software-Agenten in der Lage, Entscheidungen auf der Grundlage der aktuell gegebenen Rahmenbedingungen zu treffen und nicht solche, die vom Programmierer zur Zeit der Programmerstellung festgelegt wurden (Kamel Boulos et al. 2006, 166). Somit bieten Software-Agenten einen flexiblen Ansatz, mit sich dynamisch verändernden Rahmenbedingungen umzugehen.
- *Soziale Fähigkeit*: Agenten interagieren mit anderen Agenten oder möglicherweise Menschen über Agentenkommunikationssprachen, um bestimmte Ziele gemeinsam zu lösen. Um diese zu erreichen, kooperieren Agenten, verhandeln über Leistungen und koordinieren diese (Jennings 2001, 37).
- *Reaktivität*: Software-Agenten sind üblicherweise in eine Umwelt eingebettet und deshalb in der Lage, ihre Umwelt wahrzunehmen und auf Veränderungen mit geeigneten Aktionen flexibel zu reagieren.
- *Proaktivität*: Agenten reagieren nicht nur auf Veränderungen in ihrer Umwelt, sondern besitzen auch Ziel gerichtetes Verhalten, das sich in der Eigeninitiative für Aktionen manifestiert.

Weiterhin wird im Zusammenhang dieser Definition oft die *Mobilität* von Software-Agenten genannt (Wooldridge/Jennings 1995). Mit dieser Fähigkeit können Software-Agenten zum Verarbeitungsort von Informationen migrieren und dort z.B. umfangreiche Transformationsaufgaben durchführen. Ggf. werden nur die Ergebnisse der Verarbeitungsaufgabe zusammen mit den Agenten wieder an den ursprünglichen Ort transferiert. Mit diesem Ansatz werden Verarbeitungskapazitäten an den Speicherort der Daten verlagert und letztlich auch die Belastung der Netzwerkressourcen reduziert. Mit der Einführung von Mobilität werden aber zugleich Sicherheitsprobleme impliziert (siehe dazu auch Abschnitt 2.3.10); deshalb wird auf diese Eigenschaft im Operativbetrieb von Agentensystemen meist noch verzichtet.

2.3.1.2 Agentenbegriff mit Intelligenzaspekten

Lockemann/Nimis (2005) detaillieren den in Abschnitt 2.3.1.1 eingeführten minimalen Agentenbegriff. Dazu werden bei Agenten neun verschiedene Eigenschaften differenziert. Die notwendigen Eigenschaften von Agenten umfassen dabei die folgenden vier Aspekte (Lockemann/Nimis 2005, 29-30):

- „Ein Softwareagent ist ein in eine Umgebung *eingebettetes Rechnersystem* [Herv. durch Verf.]“ (Lockemann/Nimis 2005, 29).
- „Ein Agent erbringt einen *nützlichen Dienst* [Herv. durch Verf.]. Dieser ist gekapselt, kann also nur durch seine externe Wirkung beobachtet werden. Auf welche Weise der Dienst erbracht wird, bleibt dem Agenten überlassen“ (Lockemann/Nimis 2005, 29).
- „Ein Softwareagent erbringt seinen Dienst auf *autonome Weise* [Herv. durch Verf.]“ (Lockemann/Nimis 2005, 29).
- „Die Autonomie eines Softwareagenten ist *zielorientiert* [Herv. durch Verf.]: Das Verhalten bestimmt sich aus dem Abwägen der verschiedenen, oft widersprüchlichen Ziele, die der Agent verfolgt, und innerhalb des aktuell verfolgten Ziels aus einer Aktionsauswahl (’practical reasoning’)“ (Lockemann/Nimis 2005, 29-30).

Die beiden erst genannten Eigenschaften bestimmen, dass Softwareagenten sinnvolle Dienste erbringen, womit dieses Charakteristikum insbesondere an die Objekt- oder Komponentenorientierung (Lockemann/Nimis 2005, 30) sowie Service orientierte Architekturen (siehe dazu Abschnitt 3.2.4.2) erinnert. Auch dort werden nach einer wohl definierten Schnittstelle Dienste erbracht. Die eigentliche Dienstumsetzung ist innerhalb des Objektes, der Komponente bzw. des Dienstes gekapselt. Die übrigen der genannten Eigenschaften, die die Dienstausführung durch einen Agenten charakterisieren, fokussieren den Autonomieaspekt. Darin unterscheiden sich Dienste z.B. in einer SOA bzw. eines Agenten wesentlich. Bei ersterer ist diese Autonomie in einer nicht agentenbasierten Implementierung nicht gegeben.

Die bei *Lockemann/Nimis* (2005, 30) im Folgenden dargestellten fünf ergänzenden Eigenschaften von Agenten beschreiben insbesondere ihre Intelligenz. Intelligente Software-Agenten werden als solche verstanden, die auch komplexe und nicht deterministische Umgebungen beherrschen können (Lockemann/Nimis 2005, 30). Die Eigenschaften dieser Agenten werden nach *Lockemann/Nimis* (2005, 30) wie folgt beschrieben:

- „Ein intelligenter Softwareagent ist *reaktiv* [Herv. durch Verf.]: Er steht in ständiger Wechselwirkung mit seiner Umgebung, indem er sie laufend beobachtet und zeitgerecht auf Änderungen reagiert“ (Lockemann/Nimis 2005, 30).
- „Ein intelligenter Softwareagent bemüht sich um *Ausgewogenheit zwischen Reaktivität und Ziel-Orientierung* [Herv. durch Verf.]“ (Lockemann/Nimis 2005, 30).
- „Ein intelligenter Softwareagent kann *proaktiv* [Herv. durch Verf.] sein: Er ergreift bei der Verfolgung seiner Ziele eigene Initiativen“ (Lockemann/Nimis 2005, 30).
- „Ein intelligenter Softwareagent sollte *sozialfähig* [Herv. durch Verf.] sein, d.h. mit anderen Agenten beim Erbringen eines Dienstes zusammenarbeiten“ (Lockemann/Nimis 2005, 30).

- „Ein intelligenter Softwareagent kann *lernfähig* [Herv. durch Verf.] sein“ (Lockemann/Nimis 2005, 30).

Die eben genannten Eigenschaften von Agenten beschreiben im Gegensatz zu den in diesem Abschnitt weiter oben dargestellten Charakteristika nicht den Dienst an sich, sondern vielmehr die Qualität der Dienste, die von ihnen erbracht werden (Lockemann/Nimis 2005, 30). Insgesamt kann damit festgestellt werden, dass der minimale Agentenbegriff aus Abschnitt 2.3.1.2 durch die aufgeführten Intelligenzaspekte konkretisiert wird.

2.3.1.3 Agentenbegriff mit humanen Eigenschaften

Humane Eigenschaften können den bisher beschriebenen Agentenbegriff zusätzlich ausbauen. Darunter fallen Charakteristika wie emotionales Verhalten (Bates/Loyall/Reilly 1992; Bates 1994), Aufrichtigkeit (Galliers 1988, 159-164), Benevolenz (Wooldridge/Jennings 1995) und visuelle Darstellung z.B. über animierte Gesichter (Maes 1994, 36):

- *Emotionalität*: Mit emotionalen Agenten wird die Benutzerschnittstelle eines Software-Systems den Eigenschaften menschlicher Interaktionsmuster nachgebildet (Kirn 2002, 61).
- *Aufrichtigkeit*: Bei aufrichtigen Agenten wird angenommen, dass falsche Information von Agenten nicht bewusst weitergegeben wird (Wooldridge/Jennings 1995).
- *Benevolenz/Antagonistik*: Benevolente Agenten führen Aufträge anderer Agenten kooperativ aus, während antagonistische Agenten versuchen, konfliktorientiert auf externe Anfragen zu reagieren (Kirn 2002, 61). Weiterhin sind benevolente Agenten (Rosenschein/Genesereth 1985, 91) dadurch gekennzeichnet, dass Agenten selbst keine konfligierenden Ziele besitzen und damit so handeln, wie es von ihnen erwartet wird (Wooldridge/Jennings 1995).

2.3.1.4 Rationale Agenten

Während die bisherigen Beschreibungen von Agenten (siehe Abschnitte 2.3.1.1 mit 2.3.1.3) deren Eigenschaften darstellen, wird im Folgenden eine Implementierung über das Konzept der rationalen Agenten bzw. deliberativen Architekturen aufgezeigt.

Das Ziel eines rationalen Agenten (Galliers 1988, 49-54) ist das kontinuierliche Streben nach der Ausführung von korrekten Aktionen, wobei die Auswirkungen der Aktionen ex ante nicht in ihrer Vollständigkeit abgeschätzt werden (Braubach/Pokahr/Lamersdorf 2004, 34). Entscheidungen für die Ausführung der jeweiligen Aktionen werden vielmehr entsprechend einem angemessenen Verhältnis zwischen Aufwand und Nutzen getroffen (Kirn 2002, 60). Rationale Agenten streben damit die Erreichung von Zielen an (Wooldridge/Jennings 1995). Mit der Fokussierung auf Ziele sind Agenten in der Lage, ihre Aktionen zur Laufzeit zu planen und anzupassen (Kirn 2005, 3).

Für die Entwicklung solcher Agenten werden *deliberative Architekturen* eingesetzt (Braubach/Pokahr/Lamersdorf 2004, 34). Die deliberative Architektur eines Agenten (Wooldridge/Jennings 1995, 132) wird definiert durch die Existenz eines durch Symbole repräsentierten Modells der Realwelt (Grütter 2006, 10). Entscheidungen eines solchen Agenten, die die weiteren Aktionen bestimmen, werden auf der Grundlage von Schlussfolgerungen

getroffen, welche auf der Verarbeitung von Symbolen basieren (Grütter 2006, 10). Mit deliberativen Architekturen werden somit die Interna der Agenten, ihre Struktur sowie mögliche Ableitungen für die Zieldefinition auf der Basis von mentalistischen Konzepten bestimmt (Braubach/Pokahr/Lamersdorf 2004, 34). Der mentalistische Ansatz zeichnet sich insgesamt durch zwei wesentliche Eigenschaften aus: Er abstrahiert von der eigentlichen Implementierung, indem nur die Ziele für die Agenten modelliert werden. Weiterhin lassen sich Eigenschaften der realen Welt vorteilhaft und geradlinig auf Agenten abbilden (Braubach/Pokahr/Lamersdorf 2004, 36).

Für die deliberative Architektur eines BDI-Agenten bedeutet dies zusammenfassend, dass ein Agent somit Wissen (Belief) über sich selbst und seine Umwelt, Ziele (Desire), die zu verfolgen sind, und Pläne (Intention), die angeben, wie diese Ziele erreicht werden können, besitzt (Krempels et al. 2003, 43). Grundlagen des BDI-Ansatzes liegen in der Erarbeitung einer philosophischen Theorie des menschlichen Handelns auf der Basis von mentalistischen Ansätzen (Bratman 1987; Braubach/Pokahr/Lamersdorf 2004, 34). Diese Idee wurde bei *Rao/Georgeff* (1995) und *Kinny/Georgeff/Rao* (1996) als Grundlage für den Einsatz bei Software-Agenten übertragen.

Im folgenden Abschnitt wird beschrieben, wie die bisher dargestellten Agentenbegriffe zusammenhängen.

2.3.1.5 Zusammenhang zwischen den beschriebenen Agentenbegriffen

Der Zusammenhang zwischen den bisher erläuterten Agentenbegriffen ist in Abbildung 2.3-1 dargestellt. Dabei werden zunächst Charakteristika und Implementierungsmöglichkeiten differenziert: Die Eigenschaften von minimalen Agenten können über den Dienstbegriff von Agenten mit Intelligenzaspekten konkretisiert werden. Minimale bzw. intelligente Agenten können jeweils mit humanen Eigenschaften ergänzt werden.

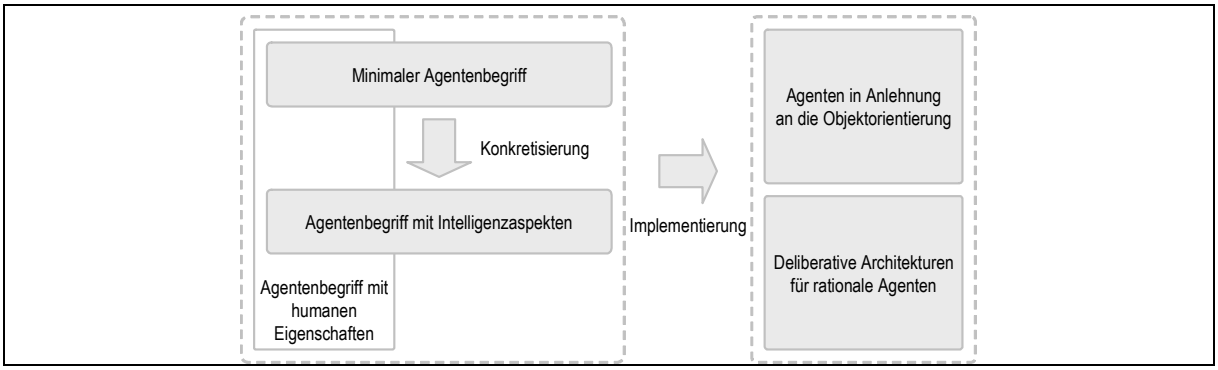

Abbildung 2.3-1: ***Zusammenhang zwischen den betrachteten Agentenbegriffen***
Quelle: Eigene Darstellung

Deliberative Architekturen für rationale Agenten stellen hingegen eine Möglichkeit der Implementierung dar. Dabei teilen die genannten Agentenbegriffe und die Implementierung durch rationale Agenten insbesondere die Zielorientierung. Eine weitere Implementierungsmöglichkeit ist durch die Anlehnung an die Objektorientierung gegeben, wie sie in der

Realisierung der vorliegenden Arbeit verfolgt wird. Als vorteilhaft sind dabei die Ausrichtung an etablierten Modellierungstechniken und Vorgehensmodellen zu nennen, womit eine breitere Akzeptanz der Ergebnisse erreicht werden kann.

Relevant für die Kategorisierung einer Software-Einheit als Agent ist nicht notwendigerweise die Erfüllung aller für die jeweilige Klasse genannten Eigenschaften (Schubert/Zarnekow/Brenner 1998). Nach Schubert/Zarnekow/Brenner (1998) können unterschiedliche Teilmengen der Agentencharakteristika eingesetzt werden, um die jeweiligen Anforderungen an das zu entwickelnde System zu erfüllen. Dementsprechend werden in der vorliegenden Arbeit diejenigen Charakteristika herausgearbeitet, die für die angestrebte Applikation relevant sind.

Die bisherige Darstellung von Agenten erlaubt abschließend eine Beschreibung einer solchen Einheit im Sinne der Dienstorientierung. Damit wird die Grundlage für den Vergleich unterschiedlicher Software-Architekturen in Abschnitt 3.3 gelegt.

2.3.2 Dienstbegriff eines Agenten

Aus den bisherigen Ausführungen in den Abschnitten 2.3.1.1 mit 2.3.1.3 kann ein Dienstbegriff wie folgt abgeleitet werden, durch den ein Software-Agent charakterisiert werden kann:

Ein Dienst, der durch einen Software-Agenten erbracht wird, kapselt Funktionalitäten über eine wohl definierte Schnittstelle. Die Dienstausführung ist durch die Eigenschaft der Autonomie gekennzeichnet. Die Qualität des erbrachten Dienstes wird durch die Reaktivität, Ausgewogenheit zwischen Reaktivität und Ziel-Orientierung, Proaktivität, soziale Fähigkeit sowie Lernfähigkeit (Lockemann/Nimis 2005, 30) bestimmt. Ergänzend kann der Dienstbegriff durch die Emotionalität, Aufrichtigkeit, Benevolenz bzw. Antagonistik optional gekennzeichnet sein.

2.3.3 Multiagentensysteme

Meist agieren Software-Agenten nicht in isolierter Form, sondern sind zu einem System zusammengefasst. Letzteres wird nach *Odell/Odine/Levy* wie folgt definiert (2004, 78):

Multiagentensysteme entstehen durch eine Ansammlung einer umfangreichen Anzahl von Software-Agenten jeweils unterschiedlicher Funktionalitäten und Fähigkeiten.

Um ein gemeinsames Ziel zu erreichen, werden Agenten zu einem Multiagentensystem (MAS) oder auch zu Subsystemen zusammengeschlossen. Die dafür erforderliche Organisationsstruktur eines Agentensystems ergibt sich jedoch nicht notwendigerweise aus expliziten Strukturierungen, sondern kann sich auch implizit durch emergentes Verhalten ergeben (Pokahr/Braubach/Lamersdorf 2005a, 300). In einem MAS existiert demnach kein zentraler Kontrollmechanismus, sondern vielmehr eine Ansammlung von dezentralen Kontrollflüssen der einzelnen Agenten. Eine Klassifizierung für die Organisationsstruktur könnte somit nach dienstorientierten Gesichtspunkten erfolgen (Pokahr/Braubach/Lamersdorf 2005a, 30). Beziehungen zwischen Agenten entstehen dabei durch das Abrufen und Anbieten von Diensten.

Aus dem fehlenden zentralen Kontrollfluss ergibt sich nun die Schwierigkeit, dass trotz der Abwesenheit einer zentralen Instanz Agenten zur Laufzeit dynamisch zur Lösung eines Problems zusammengestellt werden müssen. Um diese Zuweisung von Agenten zu einer Problemlösung realisieren zu können, müssen die einzelnen Fähigkeiten der Agenten bekannt

sein, um daraus einen kompositen Dienst zusammenstellen zu können. *Odell/Nodine/Levy* (2004) stellen ein Konzept vor, das dazu beiträgt, dass die Vorhersagbarkeit, Zuverlässigkeit sowie die Stabilität in einem auch umfangreichen Agentensystem erreicht werden können (Odell/Nodine/Levy 2004, 78). Dieser Ansatz beschreibt über die Determinanten physikalische Eigenschaften, Rollen und Gruppen einen agentenbasierten Dienstbegriff und wird im Folgenden dargestellt:

Agenten werden zunächst durch eine Erweiterung der Modellierungssprache UML® hinsichtlich ihrer physikalischen Eigenschaften (z.B. Klassifikation eines Agenten als ein JADE-Agent, für eine Darstellung von JADE siehe Abschnitt 6.3.2) und Rollen klassifiziert. Durch erstere werden die Eigenschaften und das Verhalten von Agenten determiniert. Die Rollen beschreiben die Fähigkeiten eines Agenten und die Aktivitäten, in die er involviert sein kann. Die Assoziationen zwischen Agenten werden durch die Zuweisung von Rollen bestimmt. Instanzen von Agenten werden somit einerseits durch die Klassifikation nach physikalischen Eigenschaften als auch durch die Assoziation mit Rollen bestimmt.

Eine Gruppe von Agenten wird bestimmt durch eine Menge von Agenten, welche durch ihre Rollen assoziiert sind. Anders betrachtet kann eine Gruppe als eine Komposition von Rollen verstanden werden, die jeweils mit einer beliebigen Anzahl (auch null) von Agenten assoziiert sind (Odell/Nodine/Levy 2004, 86). Die Rolle eines Agenten ist abhängig vom jeweiligen Kontext, d.h. der Assoziation mit einer bestimmten Gruppe. Agentifizierte Gruppen zeichnen sich dadurch aus, dass die gesamte Gruppe wie ein einzelner Agent angesprochen werden kann. Somit muss nach außen nicht bekannt sein, welche Agenten dieser Gruppe angehören. Die Gruppe bietet eine Schnittstelle als Summe der Rollen ihrer assoziierten Agenten. Bei nicht agentifizierten Gruppen müssen Agenten der Gruppe direkt angesprochen werden. Die Rollenzuweisung ist zusammenfassend eine ternäre Abbildung zwischen Agenten, ihren Rollen und den jeweiligen Gruppen. Bei dieser Assoziation muss nicht notwendigerweise ein Agent involviert sein. Diese leere Stelle ist dann bei Laufzeit durch einen geeigneten Agenten in seiner erforderlichen Rolle zu besetzen. Mit dieser Modellierung ist es möglich, die Komplexität und Emergenz eines Agentensystems nachzubilden und dabei insbesondere die Vorhersagbarkeit, Stabilität und Zuverlässigkeit des gesamten Systems zu erhöhen (Odell/Nodine/Levy 2004, 91).

2.3.4 Agentenplattformen

Für Agentensysteme ist eine gewisse Infrastruktur erforderlich. Diese wird als Agentenplattform bezeichnet, die eine grundlegende Laufzeitumgebung für Agenten zur Verfügung stellt. Die bedeutendsten Gremien zur Standardisierung für diese Plattformen sind die OMG™ und die FIPA (Pichler/Plösch/Weinreich 2002, 91). Die beiden daraus hervorgegangenen Architekturansätze für Agentenplattformen werden in den Abschnitten 2.3.4.1 und 2.3.4.3 dargestellt.

2.3.4.1 FIPA-Referenzarchitektur

Eine FIPA-Agentenplattform (Foundation for Intelligent Physical Agents, IEEE Foundation for Intelligent Physical Agents 2006), dargestellt in Abbildung 2.3-2, ist aus folgenden Kom-

ponenten zusammengesetzt (The Foundation for Intelligent Physical Agents (FIPA) 1998, 9-10):

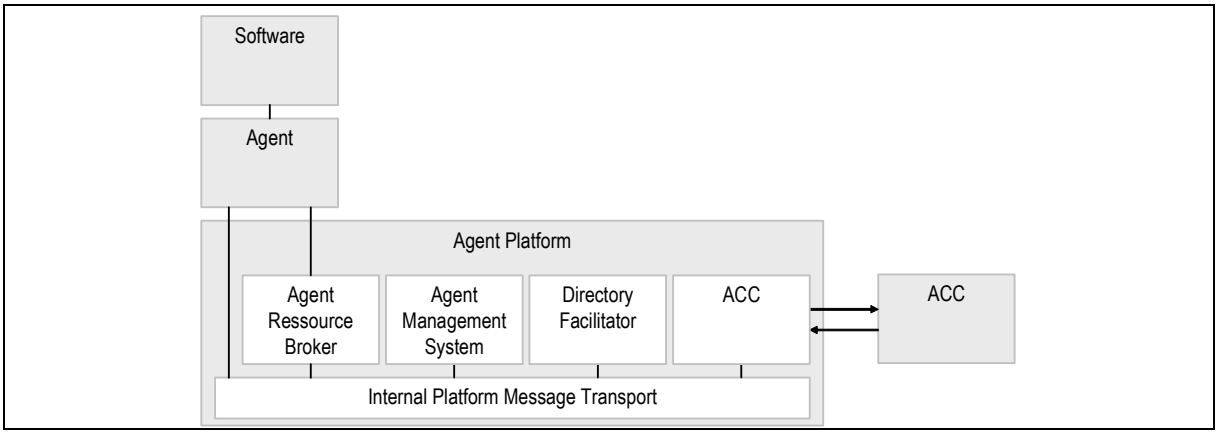

Abbildung 2.3-2: ***FIPA-Agentenplattform***
Quelle: *The Foundation for Intelligent Physical Agents (FIPA)* (1998, 9) und in Anlehnung an *Pichler/Plösch/Weinreich* (2002, 93)

- Ein *Agent* ist das wesentliche Element innerhalb einer Agentenplattform und stellt einen oder mehrere Dienste zur Verfügung, die von anderen Agenten aufgerufen werden können.
- Das *Agent Management System* (*AMS*) stellt Funktionalitäten für das Management des Lebenszyklus von Agenten bereit. Darüber hinaus enthält es ein Namensverzeichnis aller auf der Plattform registrierten Agenten (*White Pages*).
- Im *Directory Facilitator* (*DF*) sind alle Agenten mit ihren angebotenen Diensten registriert (*Yellow Pages*). Agenten können Anfragen an den DF zur Suche von bestimmten Diensten stellen.
- Für die Kommunikation mit Agenten auf anderen Agentenplattformen wird der *Agent Communication Channel* (*ACC*) eingesetzt.
- Mit der *Agent Platform* (*AP*) wird die physikalische Umgebung bereitgestellt, auf der Agenten ausgeführt werden können. Eine solche Plattform besitzt neben den bereits genannten Komponenten eine Rechenmaschine, ein Betriebssystem und weitere Software, die von Agenten benötigt wird (The Foundation for Intelligent Physical Agents (FIPA) 1998, 10).
- Mit der Komponente *Internal Platform Message Transport* (*IPMT*) wird der Nachrichtenaustausch innerhalb einer Plattform realisiert (The Foundation for Intelligent Physical Agents (FIPA) 1998, 10).
- Anwendungen, die nicht agentenbasiert entwickelt sind, werden über Wrapper-Agenten (Jennings 2001, 40) in das Agentensystem integriert. Der *Agent Ressource Broker* (*ARB*) ermöglicht das Registrieren von Diensten, die nicht von Agenten realisiert werden und ermöglicht so das Auffinden von externen Anwendungen (siehe auch Pichler/Plösch/Weinreich 2002, 93).

2.3.4.2 CORBA®

Für das Verständnis der auf CORBA aufbauenden und in Abschnitt 2.3.4.3 dargestellten MASIF-Plattform werden im Folgenden zunächst die Grundlagen von CORBA beschrieben. Dazu werden die wesentlichen Eigenschaften von CORBA dargestellt, bevor auf die Erweiterungen zur dynamischen Bindung und der ORB übergreifenden Kommunikation eingegangen wird.

CORBA (Common Object Request Broker Architecture) wurde von der OMG (Open Management Group) definiert und beschreibt eine Architektur (siehe Abbildung 2.3-3 und Abbildung 2.3-4) zur Realisierung verteilter Systeme (siehe für die folgenden Ausführungen Matyas/Maurer 2003, 96-97). Die wesentlichen Komponenten einer CORBA-Architektur sind der Object Request Broker (ORB), Aufrufschnittstellen und der Objektadapter. Aufrufschnittstellen ermöglichen die späte Bindung der Implementierung an den Methodenaufruf und verpacken die an den Empfänger zu übersendenden Argumente (Szyperski 2002, 232). Auf der Empfängerseite werden diese Argumente vom jeweiligen Objektadapter entpackt und die Methode beim adressierten Objekt aufgerufen (Szyperski 2002, 232). Das Ziel dieser Architektur ist die Transparenz und Unabhängigkeit der jeweils eingesetzten Plattform und Implementierungssprache (Pichler/Plösch/Weinreich 2002, 95).

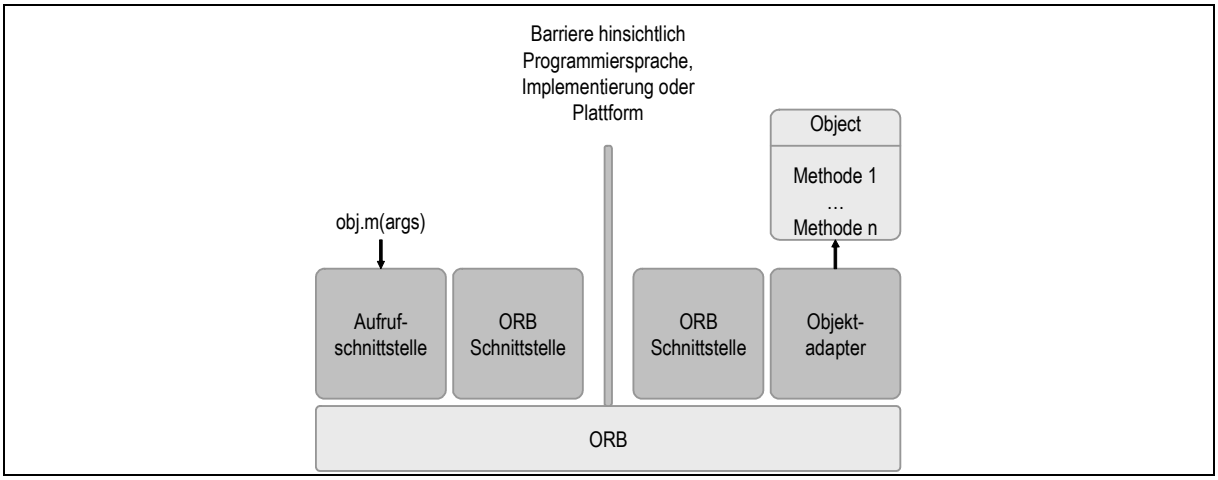

Abbildung 2.3-3: ***Wesentliche CORBA-Elemente***
Quelle: In Anlehnung an *Szyperski* (2002, 233)

Um die dynamische Bindung der Methoden auf der Server- oder Client-Seite (siehe dazu Abbildung 2.3-4) zu ermöglichen, werden dynamische Schnittstellen beim Client (Dynamic Invocation Interface) und Server (Dynamic Skeleton Interface) eingeführt (Szyperski 2002, 233). Die Schnittstellen der von entfernten Objekten definierten Dienste werden in Schnittstellendefinitionen (IDL™, Interface Definition Language) beschrieben. Skeletons und Stubs werden aus einer IDL-Beschreibung vom IDL-Compiler generiert. Diese Beschreibungen sind unabhängig von der gewählten Programmiersprache. Die angebotenen Dienste können unabhängig von der für die Implementierung des Objektes eingesetzten Programmiersprache angesprochen werden. Serverprogramme registrieren zur Laufzeit des Systems ihren Dienst über den Objektadapter im ORB. Der Client erhält ein lokales Stub-Objekt, welches die Dienste des entfernten Objektes transparent zur Verfügung stellt. Der Dienstaufruf des lokalen Ob-

jekts wird zu seiner Ausführung vom ORB an den Server weitergeleitet und dort ausgeführt. Das Skeleton empfängt die Nachricht, entpackt diese und ruft das Zielobjekt direkt auf (Szyperski 2002, 233). Ggf. werden die Ergebnisse der Berechnung über den ORB an das aufrufende Objekt zurückgeliefert.

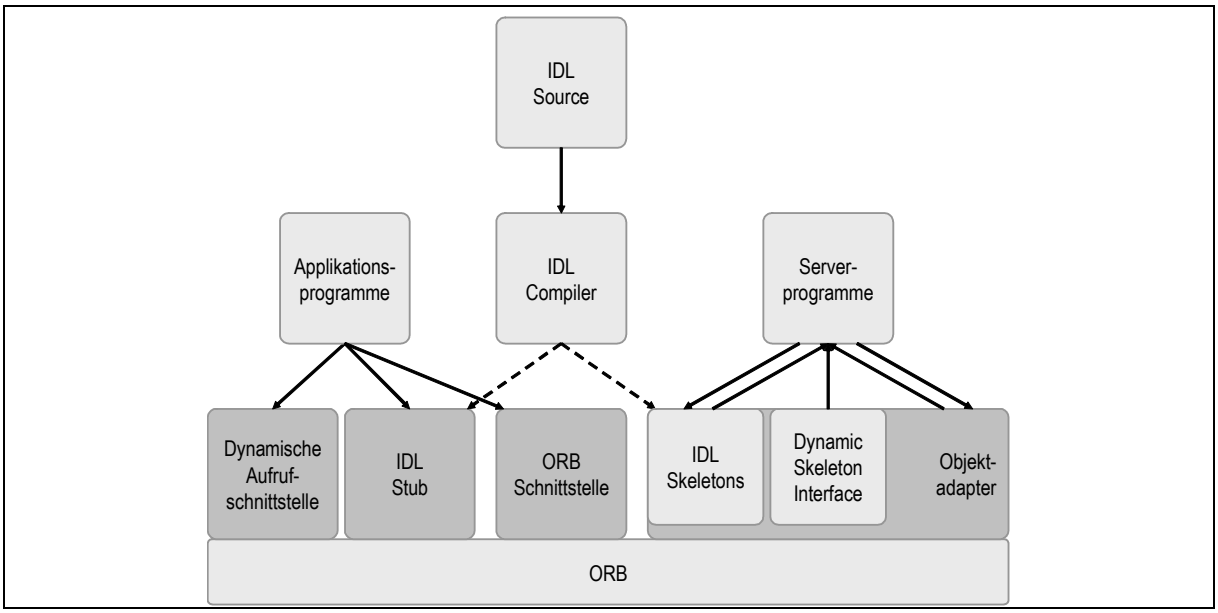

Abbildung 2.3-4: ***CORBA-Elemente bei der dynamischen Bindung***
Quelle: In Anlehnung an *Szyperski* (2002, 234)

Die Interaktion zwischen verteilten und heterogenen Objekten ist nicht nur auf einen einzigen ORB limitiert. Um die Kommunikation zwischen unterschiedlichen ORBs zu ermöglichen, wird IIOP™ (Internet Inter-ORB Protocol) definiert. Der Nachrichtenaustausch zwischen den ORBs ist standardisiert und ermöglicht so die Kommunikation zwischen heterogenen ORBs.

Obwohl auch in einer CORBA-Umgebung die von entfernten Objekten angebotenen Dienste in Analogie zu einem FIPA-Agentensystem über White bzw. Yellow Pages (siehe dazu Abschnitt 2.3.4.1) zur Verfügung gestellt werden und damit für Objekte die Möglichkeit gegeben ist, ohne die Kenntnis der Lokation von entfernten Objekten über den ORB ihre Dienste aufzurufen (Genesereth/Ketchpel 1994, 53), kann dennoch im Vergleich zu einem Agentensystem auf FIPA-Basis ein wesentlicher Unterschied identifiziert werden (Genesereth/Ketchpel 1994, 53):

Die Beschreibung der angebotenen und benötigten Dienste sowie der versandten Nachrichten kann durch die formale Kommunikationssprache ACL (siehe dazu Abschnitt 2.3.6) in einem Agentensystem in einer umfassenderen Weise erfolgen, als dies bei CORBA möglich ist. Dabei sind insbesondere die Fähigkeiten des DF wesentlich, der je nach Anforderung als Mediator in der Form eines Vermittlers, der Nachrichten transformiert, dekomponiert und kombiniert, oder lediglich zur Lokalisierung von anderen Agenten fungiert, ohne ergänzende Funktionalität zu leisten.

In dieser Hinsicht erweitert ein Agentensystem die Möglichkeiten, die durch eine CORBA-Umgebung zur Verfügung gestellt werden.

2.3.4.3 MASIF-Referenzarchitektur

Weil MASIF im Zusammenhang mit CORBA-Diensten und CORBAfacilities zu betrachten sind, werden diese im Folgenden beschrieben:

Für die Entwicklung von verteilten Anwendungen mit der CORBA-Umgebung stehen grundlegende Dienste (CORBAservices, siehe auch Abbildung 2.3-5) zur Verfügung. Diese Dienste bieten die folgenden Funktionalitäten an (für eine ausführliche Beschreibung siehe Szyperski 2002, 239-247):

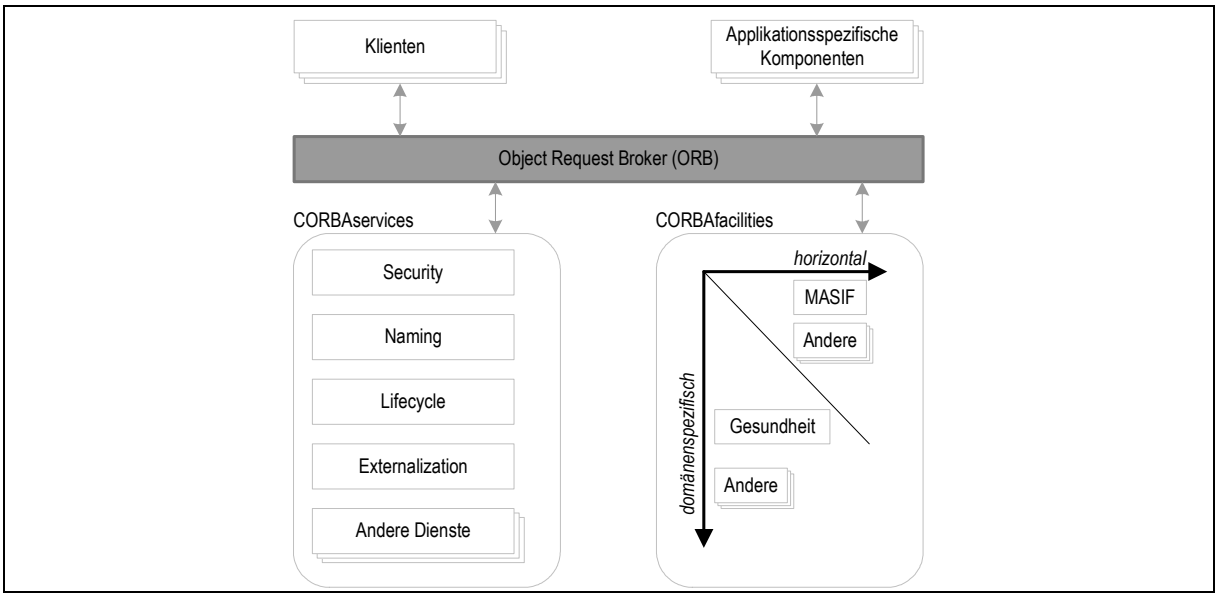

Abbildung 2.3-5: ***CORBA-Dienste sowie Zusammenhang zwischen CORBA und MASIF***
Quelle: *Pichler/Plösch/Weinreich* (2002, 95)

- Namensdienst (White Pages), Vermittlungsdienst (Yellow Pages)
- Ereignisdienst, Benachrichtigungsdienst zur Kommunikation von Ereignissen
- Transaktionsdienst
- Sicherheitsdienst zur Realisierung von z.B. Authentifizierung und sicherer Kommunikation
- Unterstützung von Nebenläufigkeit über die Verwaltung von Sperren auf Ressourcen
- Lizenzdienst zur Durchsetzung der Nutzungsbestimmungen für Komponenten
- Lebenszyklusmanagement für CORBA-Objekte
- Management von Relationen zwischen Objekten auf höherer Abstraktionsebene als durch die Realisierung einer Programmiersprache z.B. über Referenzen
- Persistenzdienst für Objektzustände
- Externalisierung zur Serialisierung und Deserialisierung von Objektgeflechten
- Funktionalität zur Zuweisung, Abfrage und Entfernung von Objekteigenschaften
- Realisierung eines Objektabfragedienstes zur Identifikation von Objekten über ihre Attributeigenschaften
- Bereitstellung von Datenstrukturen wie z.B. Listen oder Bäume für Objekte
- Zeitdienst zur Gewährleistung von uniformen Zeitstempeln

Weiterhin werden für bestimmte Anwendungsbereiche CORBAfacilities eingesetzt (Pichler/Plösch/Weinreich 2002, 95). Dabei wird zwischen zwei Dimensionen differenziert (Szyperski 2002, 256): In der vertikalen werden domänenspezifische Dienste angeboten. Zu den Domänen zählen bspw. Telekommunikation oder das Gesundheitswesen (Szyperski 2002, 257). Domänenunabhängige Dienste werden in der horizontalen Dimension bereitgestellt und umfassen Funktionalitäten zu Benutzerschnittstellen (z.B. Internationalisierung), zum Informations- (z.B. mobile Agenten), System- (Zeit) und Aufgabenmanagement (z.B. Druckfunktionalitäten, Szyperski 2002, 257).

MASIF lässt sich hinsichtlich CORBA als ein domänenunabhängiger Dienst einordnen, der CORBAservices verwendet (Pichler/Plösch/Weinreich 2002, 95). Die folgenden Ausführungen beschreiben die MASIF-Architektur mit ihren Eigenschaften (siehe dazu Pichler/Plösch/Weinreich 2002, 95):

Agenten einer MASIF-Plattform (siehe Abbildung 2.3-6) sind per se mobil und einer Autorität zugeordnet, in deren Auftrag sie ihre Aufgaben ausführen. Das Agentensystem (Agent System) stellt Managementfunktionen z.B. für den Lebenszyklus von Agenten zur Verfügung und ist mit einer Autorität assoziiert. Weiterhin besitzt das Agentensystem ein Verzeichnis über alle seine oder den mit einer bestimmten Autorität assoziierten Agenten. Ersteres kann über mehrere Orte (Places) verteilt sein. In einer Region werden alle Agentensysteme einer Autorität zusammengefasst. Weiterhin wird damit die Basis für die Rechtezuweisung an Agenten in Abhängigkeit ihrer Autorität gebildet. Im Agent Finder wird das Namensverzeichnis der Agentendienste dieser Region vorgehalten. Die Schnittstellen zum Agentensystem und Verzeichnisdienst sowie anwendungsspezifische Dienste der Agentensysteme und Agenten werden durch IDL-Beschreibungen spezifiziert.

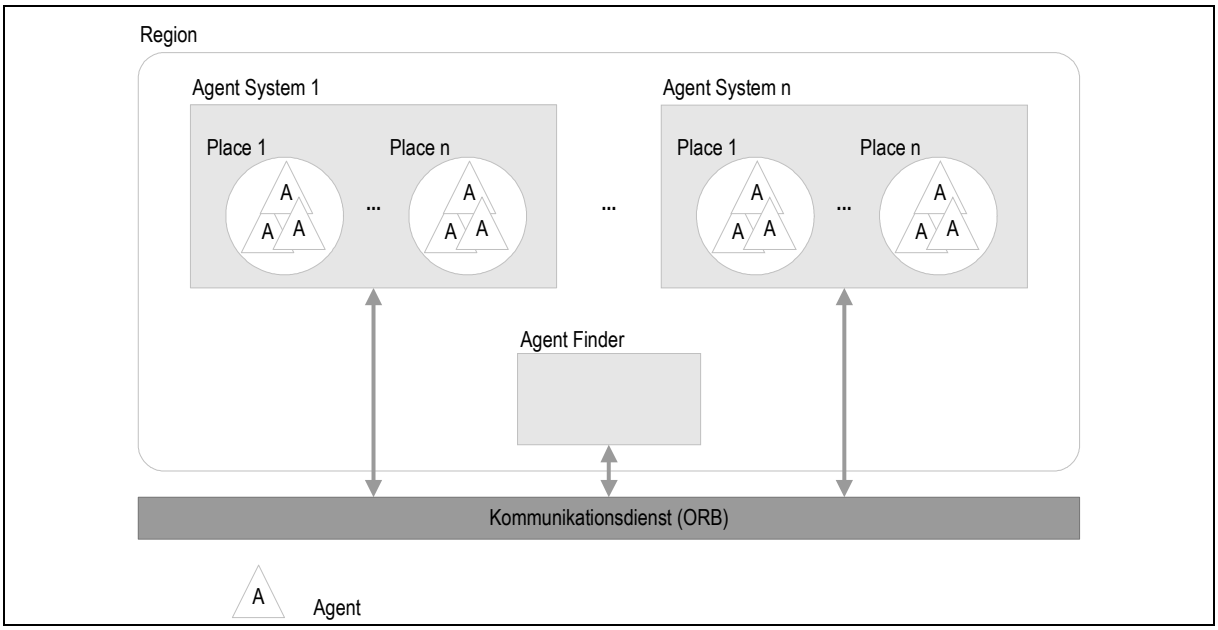

Abbildung 2.3-6: ***Darstellung der MASIF-Agentenarchitektur***
Quelle: *Pichler/Plösch/Weinreich* (2002, 95)

Der Schwerpunkt der MASIF-Architektur liegt aufgrund der Einordnung in die CORBA-Landschaft auf der Interoperabilität von Agentenplattformen auf unterschiedlichen Rechnern, wobei die eingesetzte Programmiersprache ausdrücklich nicht berücksichtigt wird (Pichler/Plösch/Weinreich 2002, 95 und 100). Damit ist die Agentenplattform lediglich in der Lage, Agenten auch auf einer anderen Agentenplattform auszuführen, wenn beide Plattformen dieselbe Programmiersprache bedingen (Pichler/Plösch/Weinreich 2002, 95-96). Wenn ein Agent als CORBA-Objekt realisiert ist, können auch die CORBA-Basisdienste von einem Agenten aufgerufen werden (Pichler/Plösch/Weinreich 2002, 96).

Im folgenden Abschnitt 2.3.4.4 werden die FIPA- und MASIF-Architekturen gegenübergestellt, um eine Entscheidung für die Auswahl einer geeigneten Plattform für die Anforderungen der vorliegenden Arbeit zu ermöglichen.

2.3.4.4 Gegenüberstellung der FIPA- und MASIF-Architektur

Mit der FIPA- und MASIF-Architektur werden unterschiedliche Ziele verfolgt. Deshalb können neben einigen Gemeinsamkeiten auch wesentliche Unterschiede identifiziert werden. Die Gemeinsamkeiten der beiden Ansätze umfassen die folgenden Aspekte (Pichler/Plösch/Weinreich 2002, 96):

- Unabhängigkeit von der einzusetzenden Implementierungssprache
- Das FIPA Agent Management System und das MASIF Agent System besitzen ähnliche Funktionalitäten.
- Ein FIPA DF ist vergleichbar mit dem MASIF Agent Finder.
- Der FIPA ACC entspricht dem ORB der MASIF-Plattform. Die Kommunikation zwischen Plattformen auf verteilten Rechnern erfolgt über IIOP. Bei FIPA ist IIOP das Basisprotokoll für die ACL-Kommunikation.

Neben den genannten Gemeinsamkeiten werden die folgenden Unterschiede identifiziert: FIPA definiert eine vollständige Infrastruktur und betrachtet die Mobilität als optional (Pichler/Plösch/Weinreich 2002, 96). Im Gegensatz dazu sind MASIF-Agenten inhärent mobil und die Infrastruktur fokussiert die Interoperabilität, womit ihr die Beschreibung von Standards für Protokolle höherer Ebenen fehlt (Pichler/Plösch/Weinreich 2002, 96). Die Erfahrung bei *Pichler/Plösch/Weinreich* (2002, 96-100) untermauert dies, indem festgestellt wurde, dass mit MASIF lediglich Schnittstellen für die Interaktion von Agentenservern definiert werden und MASIF daher als eine Erweiterung der OMG Object Management Architecture™ betrachtet werden kann (Pichler/Plösch/Weinreich 2002, 100).

Obwohl bei *Bellavista/Stefanelli* (2000) ein Rahmenwerk für die Integration von CORBA, MASIF und FIPA beschrieben wird, konnten dennoch Schwächen bei der Integration eines existierenden Agentensystems in eine CORBA-Umgebung (Pichler/Plösch/Weinreich 2002, 96-100) festgestellt werden (Pichler/Plösch/Weinreich 2002, 100).

Zusammenfassend kann konstatiert werden, dass MASIF im Gegensatz zu FIPA Interaktionsproblemstellungen fokussiert und dabei die Spezifikation einer vollständigen Infrastruktur sowie Protokolle auf abstrakter Ebene für ein Agentensystem offen lässt. Für die vorliegende Arbeit ist jedoch angestrebt, die Ergebnisse auf eine möglichst vollständige Basis einer Agentenplattform zu stellen. Weiterhin finden FIPA basierte Lösungen in der

wissenschaftlichen Gemeinschaft eine breite Unterstützung (siehe dazu Kirn et al. 2006b). Die folgenden Ausführungen fokussieren aus den genannten Gründen eine FIPA-Agentenplattform als Grundlage für die vorliegende Arbeit. Dazu werden in den folgenden Abschnitten 2.3.5 und 2.3.6 relevante Merkmale dargestellt.

2.3.5 FIPA-Lebenszyklusmodell für Agenten

Agenten, die der FIPA-Spezifikation folgen, unterliegen einem bestimmten Lebenszyklusmodell (siehe zu den folgenden Ausführungen auch The Foundation for Intelligent Physical Agents (FIPA) 1998, 13-14; Bellifemine et al. 2005, 11-12), welches in Abbildung 2.3-7 dargestellt ist:

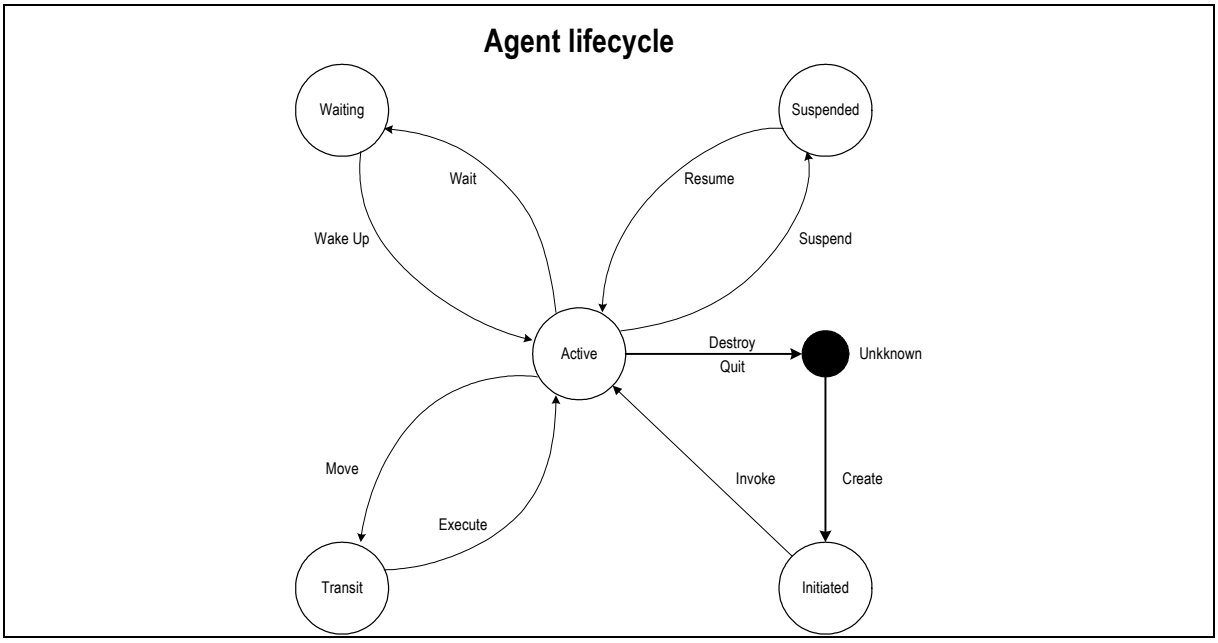

Abbildung 2.3-7: ***Lebenszyklusmodell für einen FIPA-Agenten***
Quelle: *The Foundation for Intelligent Physical Agents (FIPA)* (1998, 13)

Sobald ein Agent erzeugt wird, befindet sich dieser im Zustand „Initiated“. Damit ist zwar eine Instanz im Sinne der Objektorientierung gebildet, eine Registrierung im AMS, eine Vergabe eines Namens und einer Adresse sind jedoch noch nicht erfolgt. Dementsprechend kann noch keine Kommunikation mit anderen Agenten durchgeführt werden.

Den Zustand „Active“ erlangt ein Agent, sobald eine Registrierung im AMS erfolgte. Damit ist der Agent auch unter einer ihm zugeordneten Adresse erreichbar, besitzt einen Namen und kann mit anderen Agenten kommunizieren. In diesem Zustand werden die Behaviours des Agenten ausgeführt, durch welche die Aktivität eines Agenten determiniert wird.

Eine zwingende Terminierung („Destroy“) eines Agenten kann nur über das AMS erfolgen, wobei der Agent diesen Aufruf nicht ignorieren kann. Im Gegensatz dazu kann die Terminierung über „Quit“ vom Agenten ignoriert werden.

Der Zustand „Suspended“ wird über „Suspend“ erreicht bzw. über „Resume“ verlassen. Die Operation „Suspend“ kann von einem Agenten oder dem AMS initiiert werden, während

„Resume“ ausschließlich über das AMS angestoßen werden kann. „Suspended“ wird erreicht, wenn der interne Kontrollfluss gestoppt ist und kein Behaviour ausgeführt wird.

In den Zustand „Waiting“ kann ein Agent über „Wait“ versetzt und über „Wake Up“ wieder aus diesem in den Zustand „Active“ überführt werden. In den Zustand „Wait“ gelangen Agenten nur von sich aus, während die Operation „Wake Up“ ausschließlich vom AMS angestoßen werden kann. In dem Zustand „Wait“ wird ein Agent blockiert, weil er z.B. auf eine eintreffende Nachricht wartet. Wenn eine bestimmte Bedingung erfüllt ist, also z.B. eine erwartete Nachricht eingetroffen ist, wird wieder der Zustand „Active“ erreicht.

Nur mobile Agenten verfügen darüber hinaus über den Zustand „Transit“. Agenten initiieren diesen Zustand über die Operation „Move“, das AMS bringt einen migrierten Agenten über „Execute“ wieder in den Zustand „Active“.

2.3.6 Agentenkommunikationssprachen

Ein wesentliches Element von Multiagentensystemen ist die Kommunikation zwischen den das System bestimmenden Agenten. Diese Interaktion basiert auf einer eigenen Kommunikationssprache, welche im Folgenden beschrieben wird (Grütter 2006, 5-6):

Eine Agentenkommunikationssprache besitzt die Bestandteile äußere Sprache, innere Sprache und Vokabular (Grütter 2006, 5). In ACL-Nachrichten (Agent Communication Language) wird die Struktur des Kommunikationsmechanismus festgelegt. Man spricht dabei auch von der äußeren Sprache, welche in die beiden Bestandteile Kommunikationsschicht und Nachrichtenschicht (siehe auch Abbildung 2.3-8) untergliedert ist (Grütter 2006, 6). Erstere beschreibt die Einheiten, welche den Empfang und das Versenden der Nachrichten ermöglichen, also bspw. den Empfänger und Sender der Nachricht (Grütter 2006, 6). In der Nachrichtenschicht werden der Sprechakt und damit die Interaktionsstruktur der Kommunikation bestimmt. Die semantischen Inhalte, die eigentlichen Daten, werden in der inneren Sprache angegeben. Diese Daten entstammen einem vorab wohl definierten Vokabular, einer Ontologie (Grütter 2006, 6).

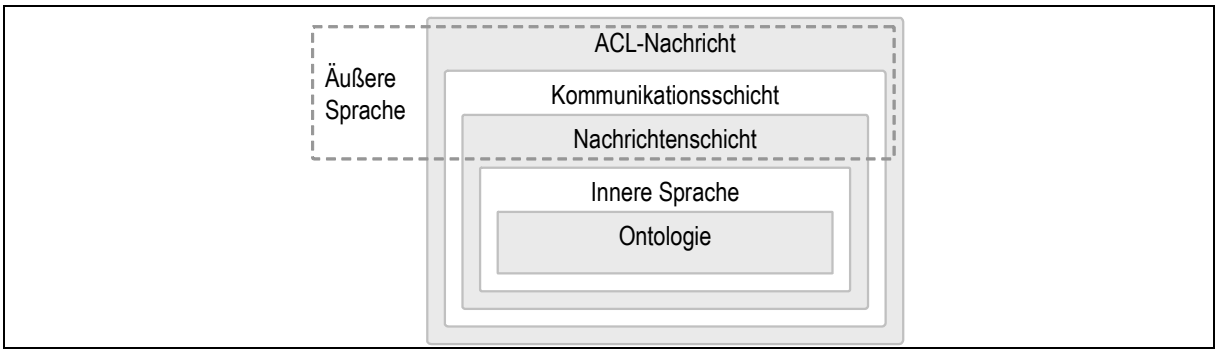

Abbildung 2.3-8: ***ACL-Nachrichtenstruktur***
Quelle: Eigene Darstellung, in Anlehnung an *Grütter* (2006, 5)

Zur Beschreibung des Inhaltes einer Nachricht wird in Agentensystemen oftmals die Syntaxspezifikation FIPA SL (Semantic Language) Content Language Specification (The

Foundation for Intelligent Physical Agents 2002) eingesetzt. Damit werden wohl geformte Terme beschrieben, die als Inhalt in eine ACL-Nachricht eingebettet werden können. Diese Terme werden als Zeichenkette in die ACL-Nachricht integriert.

Ein besonderes Kennzeichen von ACL-Nachrichten ist die bereits erwähnte Integration in definierte Interaktionsmuster. Die Gesamtheit der innerhalb einer Interaktion zwischen Agenten ausgetauschten ACL-Nachrichten bildet eine Agenteninteraktion, welche von der FIPA[9] über unterschiedliche Protokolle detailliert spezifiziert wird. ACL-Nachrichten basieren somit auf der Sprechakttheorie (Searle 1969): Performative definieren dabei einen illokutiven Akt und damit auch den weiteren Verlauf des jeweiligen Interaktionsprotokolls (Grütter 2006, 6). Illokutive Sprechakte zielen auf die vom Sender gewünschte Reaktion beim Empfänger der Nachricht ab (Grütter 2006, 6). Diese Reaktion des Empfängers wird als perlokutiver Akt bezeichnet (Grütter 2006, 6). *Searle* (1969) beschreibt insgesamt fünf verschiedene Sprechakte, die beim Empfänger der Nachricht unterschiedliche Reaktionen auslösen sollen (Schoop 1998b, 219):

- *Assertive* beschreiben Fakten der Realwelt.
- *Kommissive* beschreiben Intentionen, die der Sender der Nachricht selbst auszuführen gedenkt.
- *Direktive* beschreiben den Versuch des Senders, den Empfänger zur Ausführung der in der Nachricht enthaltenen Anforderung zu bringen.
- *Expressive* beschreiben den psychologischen Zustand des Senders gegenüber den in der Nachricht geäußerten Inhalten.
- *Deklarative* beschreiben die Veränderung eines wesentlichen Zustandes, der durch den Sprechakt letztendlich ausgelöst wird.

ACL-Nachrichten bilden zusammenfassend die Grundlage für den Austausch von Inhalten zwischen Agenten. Der Aufbau der Nachrichten und ihre Abfolge werden dabei durch ein definiertes Format bzw. ein vereinbartes Protokoll definiert. Die Nachrichteninhalte werden meist in Ontologien determiniert. Ihre Modellierung wird im folgenden Abschnitt 2.3.7 beschrieben.

2.3.7 Modellierung von Ontologien

Ontologien dienen der Modellierung eines gemeinsamen Wortschatzes in einem Agentensystem. Auf der Basis einer Ontologie können somit zwischen Agenten semantisch reichhaltige Nachrichten ausgetauscht werden, welche die Sachverhalte der Umgebung und gewünschte Aktionen der Agenten beschreiben. Trotz der semantischen Reichhaltigkeit ist die Kopplung zwischen Agenten dennoch lose, weil die inneren Nachrichteninhalte in die äußeren Nachrichten (siehe dazu die Ausführungen in Abschnitt 2.3.6) eingebettet werden. Weitere Ausführungen zur Begriffsbildung der Kopplung in verteilten Systemen sind in Abschnitt 3.4.1 dargestellt.

[9] Die Spezifikationen von FIPA-Interaktionsprotokollen können auf den folgenden Seiten abgerufen werden: http://www.fipa.org/repository/standardspecs.html, zugegriffen am 20.02.2007.

Ein in der wissenschaftlichen Gemeinschaft häufig eingesetztes Werkzeug (Kirn et al. 2006a, 205) zur Ontologiemodellierung und Integration in den Entwicklungsprozess ist Protégé[10] (Fridman Noy/Fergerson/Musen 2000). Weil auch in der vorliegenden Arbeit die Ontologie mit diesem Werkzeug erstellt wurde, wird dieses im Folgenden beschrieben. Diese Ausführungen sind an die Darstellung bei *Fridman Noy et al.* (2000) angelehnt:

In Protégé ist eine Ontologie eine Ansammlung von Klassen, Slots, Facets und Axiomen. Klassen beschreiben die Konzepte der zu modellierenden Domäne und sind damit vergleichbar mit Klassen in der Objektorientierung (siehe dazu Abschnitt 2.2.5). Zwischen Klassen können singuläre oder multiple Vererbungsassoziationen hergestellt werden, womit eine taxonomische Hierarchie gebildet wird (Fridman Noy/Fergerson/Musen 2000). Mit Slots werden, wiederum in Analogie zur Objektorientierung, die Eigenschaften oder Attribute einer Klasse beschrieben. Facets spezifizieren die Eigenschaften von Slots und Restriktionen zur Belegung ihrer Werte. Beispiele für solche Einschränkungen sind für Slots erlaubte Datentypen, Kardinalitäten oder Wertbereiche. Axiome wiederum spezifizieren weitere Einschränkungen.

In einem mehrstufigen Entwicklungsprozess für ein Multiagentensystem fließen die Ergebnisse der Domänenanalyse direkt in die Gestaltung der erforderlichen Ontologie ein (Krempels et al. 2003, 35-36). Für eine bruchfreie Transformation der Ontologie in Programmtext ist eine entsprechende Werkzeugunterstützung erforderlich. Diese Anforderung wird von Protégé erfüllt: Die modellierten Ontologien können auf JavaBeans™ (siehe auch Abschnitt 3.2.3.5) abgebildet werden (siehe dazu auch van Aart et al. 2002). Letztere können vereinfacht als Java-Klassen mit einer Sammlung von Attributen betrachtet werden, auf deren Werte über set- und get-Methoden zugegriffen wird. Protégé bildet somit Konzepte auf Klassen ab. Die Relationen zwischen den Konzepten der taxonomischen Struktur werden auf Vererbungshierarchien in Java abgebildet. Slots eines Konzeptes werden über Attribute von Java-Klassen dargestellt. Die Typen der Slots werden, falls primitive Typen vorliegen, direkt auf Java-Typen abgebildet. Bei zusammengesetzten Typen werden diese in aus Konzepten generierten Klassen transformiert. Die generierten JavaBeans können unmittelbar in der etablierten, in der vorliegenden Arbeit als Implementierungsgrundlage gewählten JADE-Agentenplattform (siehe dazu Abschnitt 6.3.2) eingesetzt werden, wodurch Brüche zwischen der Modellierung und Implementierung reduziert werden können.

Zusammenfassend lässt sich feststellen, dass über den Einsatz von Agentenkommunikationssprachen einschließlich einer Ontologie eine definierte Beschreibung von Nachrichten erfolgen kann. Damit wird, eine gemeinsame Ontologie vorausgesetzt, die Interoperabilität heterogener Multiagentensysteme unterstützt.

2.3.8 Anwendungsdomänen für agentenbasierte Systeme

Für manche Anwendungsbereiche verursacht die Implementierung eines Agentensystems einen dem hervorgehenden Nutzen nicht angemessenen Aufwand. Für geeignete Domänen können Software-Systeme jedoch vorteilhaft auf der Basis von Software-Agenten konstruiert werden. Unter dieser Prämisse können bislang nicht beherrschte Komplexität unterstützt und

[10] Das Werkzeug Protégé steht auf der folgenden Seite zur Verfügung: http://protege.stanford.edu/, zugegriffen am 11.09.2007.

Software-Systeme effizienter entwickelt werden (Jennings/Wooldridge 1998, 5-6). Für diese Anwendungsgebiete von agentenbasierten Systemen wird eine Reihe möglicher Kriterien genannt: *Kirn* (2005, 3) identifiziert folgende Eigenschaften von Anwendungsdomänen, in denen die Agententechnologie besonders Nutzen stiftend eingesetzt werden kann (siehe dazu auch Kirn 2002, 53; Shehory 2000, 77-78):

- Verteilte Systeme
- Lokale Dynamik
- Individuelle Autonomie
- Systemübergreifende Abhängigkeiten, intra- sowie interinstitutionaler Natur, bestimmen das Verhalten des Gesamtsystems

Weiß (2002) ergänzt und detailliert diesen Kriterienkatalog und beschreibt, dass agentenbasierte Systeme dann potenziell vorteilhaft eingesetzt werden können, wenn eine Vielzahl interagierender Komponenten existiert, die zusätzlich die folgenden Eigenschaften besitzen (Weiß 2002):

- Die interagierenden Komponenten sind ex ante nicht bekannt und können während der Laufzeit zum Gesamtsystem hinzugefügt oder von ihm entfernt werden (i.e. lokale Dynamik im Sinne von Kirn 2002, 53).
- Eine vollständige Steuerung ihres Verhaltens ist nicht möglich oder gewollt (d.h. individuelle Autonomie im Sinne von Kirn 2002, 53).
- Die Komponenten müssen über Kommunikation und Koordination interagieren, um ihre eigenen oder gemeinsame Ziele zu erreichen (d.h. in einem verteilten System erforderliche Kommunikation im Sinne von Kirn 2002, 53).

Zusammenfassend lässt sich feststellen, dass Agentensysteme in denjenigen Bereichen besonders vorteilhaft eingesetzt werden können, in denen Flexibilität erforderlich ist (Yu/Cysneiros 2002; Kirn 2006): Auf der Grundlage der Gesamtheit der genannten Kriterien werden bei *Weiß* (2002) als beispielhafte Anwendungsdomänen u.a. Geschäftsprozessmanagement (siehe dazu auch Benmerzoug/Boufaida/Boufaida 2004) oder Logistik genannt. Software-Agenten wird außerdem zur Geschäftsprozessintegration ein wesentlicher Lösungsbeitrag (Pokahr/Braubach/Lamersdorf 2005a, 304) beigemessen. Weiterhin können Software-Agenten die Integration von Altsystemen unterstützen, indem sie über dedizierte Wrapper-Agenten andere Informationssysteme in Agentensysteme integrieren (Jennings 2001, 40). Zudem können Agenten beim Aufbau von virtuellen Organisationen einen wesentlichen Mehrwert leisten (Berger/Bauer/Watzke 2001, 354), indem Prozesse automatisiert und intelligentere Dienste angeboten werden, die z.B. die Personalisierung oder den Umgang mit zunehmender Informationsmenge unterstützen (Berger/Bauer/Watzke 2001, 354). Konkret werden Vorteile für Agentensysteme in den folgenden Bereichen identifiziert (Jennings/Sycara/Wooldridge 1998, 25-30; Jennings/Wooldridge 1998, 11-17), wobei jeweils Beispiele für implementierte Systeme angegeben werden:

- Prozesssteuerung[11]

[11] *Corera/Laresgoiti/Jennings* (1996), *Jennings et al.* (1996b), *Perriollat et al.* (1996), *Mönch* (2006).

- Fertigungssteuerung (Woelk et al. 2006)
- Telekommunikation (Griffeth/Velthuijsen 1994)
- Flugsicherung (Ljungberg/Lucas 1992)
- Verkehrsmanagement (Burmeister/Haddadi/Matylis 1997)
- Informationsmanagement (Informationsfilterung und Informationssuche, Lieberman 1999; Sycara 1999; Wiesman et al. 2006)
- Electronic Commerce (Chavez/Maes 1996)
- Geschäftsprozessmanagement (Jennings et al. 1996a)
- Gesundheitswesen (für einen Überblick über bisherige Ansätze siehe Abschnitt 5.5)
- Unterhaltung (Computerspiele, Wavish/Graham 1996; interaktives Theater und Kino, Hayes-Roth et al. 1990)
- Medizinische Applikationen (für eine Beschreibung siehe Abschnitt 5.5)

2.3.9 Verbreitung von Agentensystemen

Agentensysteme werden entsprechend den Ausführungen in Abschnitt 2.3.8 besonders bei der Konstruktion von verteilten und komplexen Software-Systemen als vorteilhaft betrachtet (Jennings 2001), insbesondere in bestimmten Anwendungsdomänen. Dennoch befinden sich Multiagentensysteme bis dato nicht weit verbreitet im Einsatz (Yu/Cysneiros 2002; Heine 2005, 20), trotz der in Abschnitt 2.3.8 angegebenen, teilweise bereits eingesetzten Systeme. Als Gründe hierfür können die fehlende Robustheit von Agentenplattformen sowie nicht vorhandene, geeignete Entwicklungsmethoden identifiziert werden[12]. Weitere Hinderungsgründe werden in der Existenz unterschiedlicher Konzepte als Grundlage für die Entwicklung von Modellen für Agenten und die Diversifizierung von für ihre Realisierung notwendigen Plattformen identifiziert (Pokahr/Braubach/Lamersdorf 2005a, 305). Auch bei den Vorgehensmodellen zur Entwicklung eines Agentensystems kann ein breites Spektrum unterschiedlicher Ansätze (für eine Auswahl siehe Abschnitt 1.7) gefunden werden (Pokahr/Braubach/Lamersdorf 2005a, 305).

Diesen Diversifizierungen stehen die folgenden Standardisierungsbemühungen gegenüber: Mit der FIPA (IEEE Foundation for Intelligent Physical Agents 2006) existiert ein IEEE®-Standardisierungsgremium, das die Grundlagen für die Interoperabilität mit anderen Technologien und die Agententechnologie (sieh Abschnitt 2.3.4.1) erarbeitet. Für den Bereich der Agentenplattformen ist in diesem Zusammenhang noch MASIF von OMG (siehe Abschnitt 2.3.4.3) zu nennen. In der wissenschaftlichen Gemeinschaft setzte sich FIPA durch.

Mit der AUML (Agent UML, Bauer/Müller/Odell 2001; FIPA Modeling TC 2004) entstand ein dediziertes Pendant zur UML für die Modellierung von Agenten. Damit konnte sich UML auch im Bereich der Agenten als Modellierungssprache etablieren. AUML wird jedoch zum gegenwärtigen Zeitpunkt nicht mehr weiterentwickelt, weil Elemente davon in UML 2.0 bzw. ihre Weiterentwicklungen (Object Management Group 2007b) aufgenommen wurden. Somit

[12] *Sycara* (1998, 89), *Cernuzzi/Rossi* (2002), *Yu/Cysneiros* (2002), *Dam/Winikoff* (2003), *Luck et al.* (2004, 20), *Heine* (2005, 25-26).

konnte auch die herkömmliche Software-Entwicklung von den Ergebnissen der Agentenforschung profitieren. Die Konstruktion von agentenbasierten Systemen mit UML 2.0 bzw. mit einem erweiterten Metamodell von UML 2.0 wird bei *Bauer/Odell* (2005) bzw. *Torres da Silva/De Lucena* (2007) beschrieben. Damit kann eine Vereinheitlichung der Modellierungsmöglichkeiten bei der Software-Entwicklung beobachtet werden. Erstere ist eine wesentliche Voraussetzung für eine weitere Verbreitung von Agentensystemen in praktischen Anwendungsfeldern (Yu 2001a).

2.3.10 Sicherheit in Agentensystemen

Verteilte Systeme sind aufgrund ihrer inhärenten Eigenschaft besonderen Sicherheitsrisiken ausgesetzt. *Anderson* nennt für verteilte Systeme insbesondere Angriffe auf der Ebene von Netzwerkprotokollen, in Form von Trojanern, Viren und Würmern sowie das unberechtigte Eindringen in Netze (2001, 367-390). In herkömmlichen verteilten Systemen werden diese Angriffe durch geeignete Mechanismen wie z.B. Firewalls verhindert.

Die durch Software-Agenten induzierten Sicherheitsgefahren ähneln denjenigen, die üblicherweise in verteilten Systemen identifiziert werden können. Dennoch sind in agentenbasierten Systemen weitergehende Maßnahmen erforderlich, um z.B. die Interaktion über Firewalls hinweg und damit die Kommunikation zwischen migrierenden Agenten und den sie beherbergenden Hosts zu ermöglichen. In einem Agentensystem sind deshalb Sicherheitseigenschaften aus einer besonderen Perspektive zu betrachten, aus der sich nach *Kamel Boulos et al.* (2006, 166) die folgenden Herausforderungen ergeben:

1. Ein mobiler Agent ist Sicherheitsgefahren durch den Host ausgesetzt, auf den der Agent gerade migiert.
2. Der Host ist durch den mobilen Code eines Agenten Sicherheitsrisiken ausgesetzt.
3. Authentifikationsmechanismen sind durch die eingesetzten Techniken begrenzt.
4. Authorisierung sowie ihre Implementierung sind bisher noch wenig exploriert.
5. Firewalls können einen Host vor bösartigem Code eines mobilen Agenten schützen. Damit kann jedoch legitime und manchmal sogar erforderliche Kommunikation unterbunden werden.
6. Wie kann gewährleistet werden, dass die Ergebnisse der Berechnungsoperationen keine individuellen Datensätze enthüllen, die vom Hostsystem nicht nach außen gegeben werden dürfen? Wie kann sichergestellt werden, dass die Daten, die vom Agenten aggregiert werden, diesem und nur diesem zugeordnet werden?

Um eine Lösung für diese Problemstellungen zu erarbeiten, beschreiben *Kamel Boulos et al.* (2006, 166-168) einen Ansatz auf der Basis von virtuellen Organisationen, der im Folgenden, angelehnt an diese Darstellungen, beschrieben wird: *Kamel Boulos et al.* (2006, 167) unterscheiden zunächst eine Bandbreite von Definitionen für den Terminus der virtuellen Organisation: Ein Extrem definiert dabei virtuelle Organisationen als die Emergenz oder Spezifikation von normbasierten Systemen. Diese Normen beschreiben das Verhalten der an der Organisation teilnehmenden Einheiten. Nach außen bildet eine solche virtuelle Organisation ein durch die Teilnehmer bestimmtes Verhalten ab. Das andere Extrem beschreibt virtuelle Orga-

nisationen als eine Ansammlung von Elementen, die sich auf die Erreichung eines gemeinsamen Ziels festlegen, das jedes Element von sich aus nicht in der Lage wäre zu erreichen. In der Mitte dieses Spektrums, an dessen Enden die genannten Extrema angesiedelt werden, rangieren virtuelle Organisationen, deren Komponenten keine individuellen Akteure sind, aber Teil einer Organisation und damit durch die dort geltenden Regeln in ihrem Verhalten eingeschränkt sind.

Die Bildung einer virtuellen Organisation im Zusammenhang mit Agenten (siehe auch Kamel Boulos et al. 2006, 166) ist in Abbildung 2.3-9 dargestellt. Dabei werden Daten aus den Organisationen A und B in der virtuellen Organisation aggregiert. Ein mobiler Agent kann zu dieser Organisation migrieren, dort die gewünschten Berechnungen durchführen und mit den Ergebnissen zum Ausgangspunkt zurückkehren. Die in der virtuellen Organisation gespeicherten Daten obliegen den Regeln, wie sie innerhalb derjenigen Organisation existieren, aus der die Daten stammen.

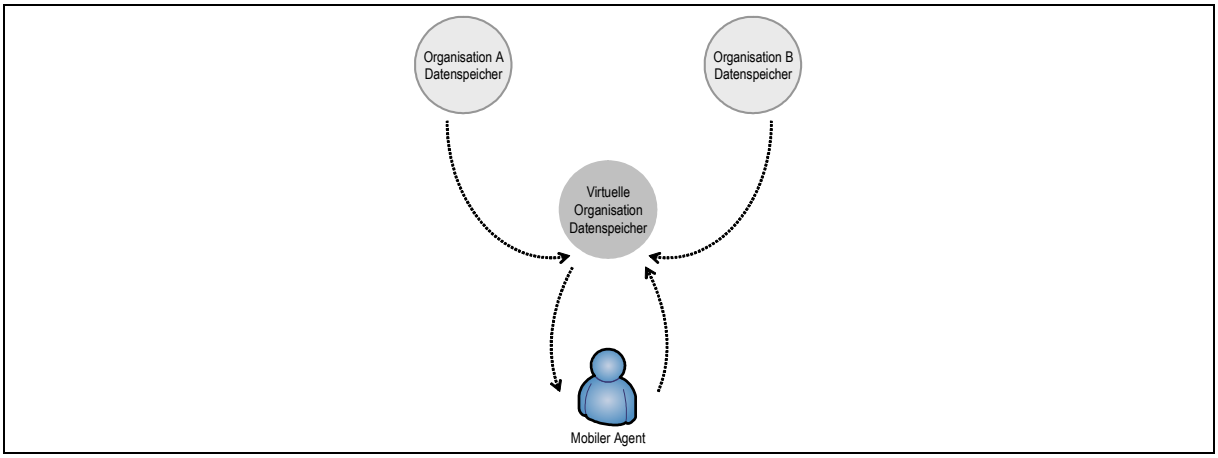

Abbildung 2.3-9: ***Bildung einer virtuellen Organisation***
Quelle: In Anlehnung an *Kamel Boulos et al.* (2006, 167)

Die Lösung der mit der Mobilität von Agenten assoziierten Problemstellungen (siehe weiter oben in diesem Abschnitt) wird von *Kamel Boulos et al.* (2006, 167-168) wie folgt beschrieben: Die Vorteile einer virtuellen Organisation liegen in der Tatsache, dass die Host-Organisationen gegenüber den Agenten geschützt sind und der Zugriff auf die Daten aus der jeweiligen Organisation entsprechend den dort geltenden Regeln forciert wird. Somit werden die Problemstellungen 2 und 4 gelöst. Aspekt 6 wird erfüllt, weil die Ergebnisse der Agentenberechnung in der virtuellen Organisation nicht den ursprünglichen Organisationen (im betrachteten Beispiel Organisationen A und B) zur Verfügung gestellt werden und weiterhin im Besitz des aggregierenden Agenten sind. Problemstellungen 1 und 5 können durch den Einsatz von Proxy-Agenten gelöst werden, welche im Auftrag des Agenten zur virtuellen Organisation migrieren und die dort die durch das Regelwerk der ursprünglichen Organisationen bestimmten Aktionen durchführen können. Problemstellung 3 wird durch den Einsatz ausreichender Mechanismen zur Authentifizierung gegenüber einem anderen Rechnersystem gelöst. Somit kann festgestellt werden, dass mit dem Ansatz einer virtuellen Organisation wesentliche Gefahrenpotenziale beseitigt werden können, die sich im Zusammenhang mit mobilen Agenten ergeben.

Obwohl in der vorgeschlagenen Lösung von Kamel Boulos et al. mobile Agenten im eigentlichen Sinne nicht verwendet werden, ist der Einsatz mobiler Agenten prinzipiell möglich (2006, 168). Dazu ist aber die Weiterentwicklung entsprechender Sicherheitsmechanismen erforderlich, welche zum aktuellen Zeitpunkt nicht durch Laufzeitumgebungen unterstützt werden (Kamel Boulos et al. 2006, 168).

Weitere Ansätze zur Gewährleistung von Sicherheit werden in Abschnitt 6.3.4 bzw. bei Roth/Jalali (2001), Pinsdorf et al. (2002), Binder/Roth (2002), Roth (2002) und Peters (2006) beschrieben. Weil alle diese Ansätze jedoch keine JADE-Erweiterung darstellen, welche die für die vorliegende Arbeit ausgewählte Agentenplattform JADE (siehe dazu Abschnitt 6.3.2) per se unterstützt, bzw. noch Defizite aufweisen (Kamel Boulos et al. 2006), werden diese in der vorliegenden Arbeit für die prototypische Implementierung nicht berücksichtigt. Stattdessen werden Mechanismen (siehe dazu Abschnitt 6.3.4) eingesetzt, die sich unmittelbar in die JADE-Plattform einbinden lasen.

Die bisherigen Ausführungen zu Programmiertechniken und Software-Agenten dienen dem im folgenden Abschnitt dargestellten Vergleich mit der Objektorientierung. Somit werden die wesentlichen Unterschiede zwischen der als etabliert geltenden Objektorientierung und der Agententechnologie beschrieben. Dabei ist zu beachten, dass die Objektorientierung wesentliche Eigenschaften derjenigen Programmiertechniken einschließt, welche in der vorliegenden Arbeit beleuchtet wurden, und diese um methodische Anreicherungen ergänzt.

2.3.11 Agenten und Objektorientierung

Um Software-Agenten in den Zusammenhang mit den in den Abschnitten 2.2.2 mit 2.2.5 beschriebenen Programmiertechniken stellen zu können, wird im Folgenden monolithische, modulare, objektorientierte sowie agentenorientierte Programmierung analysiert. Diese Programmiertechniken werden dazu aus den Perspektiven des Verhaltens, Zustandes und Funktionalitätsaufrufs jeweils einer Programmeinheit betrachtet (siehe Tabelle 2.3-1). Weiterhin wird ein detaillierter Vergleich zwischen der Objektorientierung und Agenten angestellt. Die folgenden Ausführungen zum Vergleich der Programmiertechniken sowie die Herausarbeitung der Unterschiede zwischen Agenten und Objekten sind die Zusammenfassung der Darstellung bei *Odell* (2002, 41-46):

Programmiertechnik / Kriterien	Monolithisches Programm	Modulare Programmierung	Objektorientierte Programmierung	Agentenorientierte Programmierung
Verhalten einer Einheit	Nicht modular	Modular	Modular	Modular
Zustand einer Einheit	Extern	Extern	Intern	Intern
Aufruf einer Einheit	Extern	Extern (wird aufgerufen)	Extern (über das Empfangen einer Nachricht)	Intern (Regeln, Ziele)

Tabelle 2.3-1: ***Entwicklung der Programmiertechniken***
Quelle: In Anlehnung an *Parunak* (1997, 69) und *Odell* (2002, 41)

Bei einem monolithischen Programm ist dieses selbst per se die elementare Einheit, wobei der Programmierer die vollständige Kontrolle über dieses hat. Der Zustand ist ebenso durch den Programmierer bestimmt, wobei ein solches Programm durch den Systemadministrator aufgerufen wird. Modularität kann einem solchen Programm nicht attribuiert werden, weil es nicht für unterschiedliche Zwecke wiederverwendet werden kann.

Umfangreiche Programme erfordern eine adäquate Modularisierung, um ihre Komplexität beherrschbar zu gestalten. Die Einführung der prozeduralen Programmierung bzw. ihrer Prozeduren erlaubt diese Strukturierung von Programmen. Prozeduren stellen dabei die elementare Einheit dar, in der Berechnungsvorschriften zusammengefasst sind. Das Verhalten ist damit in einer Prozedur gekapselt. Die für die Verarbeitung relevanten Daten werden von extern über die aktuelle Parameterliste der Signatur übergeben bzw. als globale Variablen referenziert. Damit wird der Zustand einer Verarbeitungsvorschrift extern gesteuert. Die Berechungsvorschriften einer Prozedur werden ausgeführt, sobald diese von außen aufgerufen wird.

Die modulare Einheit in der Objektorientierung bilden Objekte. In diesen sind Attribute sowie Methoden und damit auch ihr jeweiliges Verhalten gekapselt. Der Zustand eines Objektes ist gegeben durch die aktuelle Belegung der Attribute. Die Veränderung dieser Attribute wird nur durch Methoden vorgenommen, eine direkte Manipulation von Attributen ist in einer strikten Implementierung entsprechend dem objektorientierten Paradigma nicht vorgesehen. Somit wird der Zustand eines Objektes durch die erlaubten Operationen einer Methode intern gesteuert. Um Methoden aufzurufen, werden externe Nachrichten an das Objekt geschickt. Diese Nachrichten zwischen Objekten sind über die Definition der Methodensignatur typisiert.

Agenten sind in der Agentenorientierung die modulare Einheit. Sie besitzen einen eigenen Kontrollfluss, dem nicht nur die internen Daten bzw. ihr Zustand und das Verhalten, sondern auch der Aufruf ihrer ausführbaren Dienste unterworfen sind (Wooldridge 1997). Mit der Existenz von eigenen Regeln und Zielen werden Agenten zu aktiven Objekten. Agenten sind autonom, weil sie entsprechend ihren eigenen Zielen handeln können. Die Autonomie erlaubt Agenten zu entscheiden, wie und ob sie auf empfangene Nachrichten reagieren. Hierin ist der wesentliche Unterschied hinsichtlich des Aufrufes einer Einheit im Vergleich zu den anderen, in diesem Abschnitt genannten Programmiertechniken gegeben.

Die bisherigen Ausführungen in diesem Abschnitt zeigen, dass sich Agenten wesentlich von herkömmlichen Programmiertechniken unterscheiden. Im Folgenden werden die besonderen Differenzierungsmerkmale von Agenten und der Objektorientierung beschrieben:

Agenten besitzen interaktive Eigenschaften, die sich im Aufruf von Methoden, aber auch im Austausch von semantisch reichhaltigen Nachrichten manifestieren. In diesem Zusammenhang wird bei der Autonomie von Agenten zwischen den Aspekten Dynamik und Nichtdeterminismus differenziert (ausführlich dargestellt bei Odell 2002, 42-45). Die dynamische Autonomie reicht auf einer Skala von passiv über reaktiv bis zu proaktiv. Agenten können einerseits auf Methodenaufrufe reagieren, andererseits können sie ihre Umwelt auf Veränderungen überprüfen, geeignete Aktionen durchführen und sind damit proaktiv. Im Vergleich dazu sind Objekte streng passiv oder reaktiv. Die Passivität ist mit dem Aufruf von Methoden über einen zentralen Kontrollfluss einer anderen Instanz begründet. Die Ereignisbasierung zur Verarbeitung von Benutzerinteraktionen erlaubt Objekten einen reaktiven Status. Bei Agenten ist es im Gegensatz zu Objekten nicht nur optional, sondern meist üblich, dass sie nicht deterministisches Verhalten besitzen. Nichtdeterminismus entsteht neben der Entscheidung, wie auf eingehende Nachrichten reagiert wird, durch das asynchrone Kommunikationsmodell bei Agenten. Daraus folgt, dass der Kontrollfluss von einem zum anderen Agenten nicht ex ante

bestimmt ist. Die parallele Verarbeitung durch einen jeweils eigenen Kontrollfluss in einem Agenten bringt weiteren Nichtdeterminismus in ein Agentensystem.

Hinsichtlich der Interaktivität kann zwischen einfachen Methodenaufrufen mit einer wohl definierten Signatur, reaktivem Verhalten und einer sozialen Interaktion unterschieden werden. Letztere erlaubt die mehrfache und parallele Interaktion mit anderen Agenten. Bei Agenten wird die Kommunikation durch eine Agentenkommunikationssprache erreicht. Diese Kommunikationsform kann bei Agenten im Vergleich zu Objekten mehrere gleichzeitige Interaktionen umfassen, weil die Nachrichten mit Identifikatoren den jeweiligen Interaktionen zugeordnet werden können. In der Objektorientierung enthalten Methodenaufrufe die genaue Spezifikation der Aufrufschnittstellen.

Im Folgenden werden weitere Unterscheidungsmerkmale zwischen Agenten und Objekten beschrieben (Odell 2002, 47-49). Dabei muss ein Agentensystem aber nicht alle diese Eigenschaften besitzen bzw. kann ein objektorientiertes System dennoch einige dieser Merkmale nachbilden (Odell 2002, 47).

- *Dezentrale Steuerung* (Odell 2002, 47): In ihrer Reinform können objektorientierte Applikationen als zentrale Systeme betrachtet werden, weil sie über einen zentralen Kontrollfluss gesteuert werden. Agenten hingegen werden als dezentrales System aufgefasst (Wooldridge 1997; Jennings/Sycara/Wooldridge 1998, 15). In einer voraussagbaren und stabilen Umgebung eigenen sich deshalb zentrale, bei nicht voraussagbaren und instabilen Umgebungen eher dezentrale Systeme.
- *Multiple und dynamische Klassifikation* (Odell 2002, 47-48): Objekte sind außer im Fall der multiplen Vererbung Instanz nur einer Klasse. Diese Eigenschaft kann zur Laufzeit nicht mehr verändert werden. Agenten hingegen können zum gleichen oder zu unterschiedlichen Zeitpunkten mehrere verschiedene Rollen einnehmen und auch wieder ablegen. Diese Rollen können auch abhängig von der aktuellen Domäne sein, in der sich ein Agent gerade befindet.
- *Dynamische Änderung von Eigenschaften* (Odell 2002, 48): Die Eigenschaften von Objekten sind vorgegeben durch die jeweilige Klasse. Diese Eigenschaften sind statisch, es können keine weiteren Eigenschaften hinzugefügt werden. Agenten hingegen können ihr Verhalten z.B. durch Lernmechanismen ändern. Eigenschaften können somit erweitert, aber auch reduziert werden.
- *Bedeutung in einem System* (Odell 2002, 48): Fällt eine bestimmte Funktionalität eines Objekts in einem objektorientierten System aus, tritt meist eine Ausnahme auf. In Agentensystemen hingegen kann die gesamte Funktionalität weiterhin erbracht werden, weil Agenten zu einem Ergebnis jeweils nur einen Anteil beisteuern und ihre Berechnungsresultate hinsichtlich der Gesamtleistung nahezu vernachlässigbar sind.
- *Einschränkung der Perzeption* (Odell 2002, 48-49): Agenten können nur ihre lokale Umwelt wahrnehmen. Zur Erreichung ihrer Ziele ist diese lokale Wahrnehmung ausreichend. Die Auswirkungen ihrer lokalen Aktionen sind aber global bemerkbar. Auch mit Objekten kann diese Betrachtung der lokalen Einschränkung umgesetzt werden, wenn Objekte nur mit den mit ihnen assoziierten Objekten kommunizieren. Bei Agenten ist diese Einschränkung der Perzeption jedoch inhärent und wird üblicherweise so verwendet.

- *Emergenz* (Odell 2002, 49): Emergentes Verhalten entsteht durch die Kooperation einer Gruppe von Agenten, die sich nach außen wie eine einzige Komponente verhalten. Obwohl keine zentrale Steuerung vorhanden ist, wird dennoch das Ziel mit einem emergenten Verhalten erreicht. Weil Objekte meist über eine zentrale Kontrolle gesteuert werden, kann sich dabei kein emergentes Verhalten entwickeln.
- *Analogien aus der Natur* (Odell 2002, 49): Multiagentensysteme sind im Gegensatz zu Objekten an eine Vielzahl an Konzepten aus der Natur angelehnt.

Zusammenfassend kann festgestellt werden, dass im Vergleich zur Objektorientierung die Agentenorientierung einen höheren Abstraktionsgrad hinsichtlich des Verhaltens und der Charakteristika von Systemen (Yu 2001b, 123) besitzt. Weiterhin kann mit der Agentenorientierung eine über die Objektorientierung hinausgehende Verteilung von Systemen erzielt werden (Kirn 2002, 55). Die Herausarbeitung dieser Vorteilhaftigkeit bei der Anwendung der Agententechnologie im Bereich verteilter Systeme im Gesundheitswesen ist wesentliches Ziel der vorliegenden Arbeit. Da diese Anforderungen aber nicht für jede Systementwicklung gegeben sind, ist in ihrer Abhängigkeit zu eruieren, welches Paradigma geeignet ist.

2.4 Zusammenfassung

Die Entwicklung der unterschiedlichen Programmiertechniken in der Informatik zeigt eine zunehmende Abstraktion und Modularisierung. Agentensysteme erlauben eine über die Abstraktion von Maschinenarchitekturen hinausgehende Abstraktion (Yu 2001a), die sich auf Konzepte der Domäne stützt (Jennings 2001, 39). Die Spezifikation eines Softwaresystems kann mit dem Einsatz des Agentenparadigmas durch die Beschreibung des Agentenverhaltens geradliniger erfolgen, als dies für objektorientierte Vorgehensweisen der Fall ist (Jennings 2001, 40; Kamel Boulos et al. 2006, 165). Es werden dazu oftmals deklarative Hochsprachen eingesetzt, die nur die Ziele für den Agenten, aber nicht den Weg zur Erreichung der Aufgaben beschreiben (Kamel Boulos et al. 2006, 165). Agenten bestimmen auf der Grundlage ihrer Zielbeschreibung, ob sie mit bestimmten Aktionen ihrem Ziel näher kommen (Kamel Boulos et al. 2006, 165). Eine andere Möglichkeit besteht in der Darstellung der Entscheidungsfindung über Logikprogrammierung (Kamel Boulos et al. 2006, 165).

Die Modularisierung erlaubt die Beherrschung der Komplexität von umfangreichen Software-Systemen durch Strukturierung und Wiederverwendung. Diese Mechanismen sind bei herkömmlichen Software-Systemen jedoch oft auf Komponenten der Subsystemebene, d.h. Entwurfsmuster oder Komponenten im Sinne der Komponentenorientierung, beschränkt (Jennings 2001, 39). Agentensysteme hingegen erlauben die Wiederverwendung von vollständigen Subsystemen (Jennings 2001, 39).

Mit der Modularisierung wird auch die Kapselung von Daten hinter einer Schnittstelle angestrebt (Parunak 1999). Agenten besitzen zusätzlich ihren eigenen Kontrollfluss und kapseln damit auch die Auswahl von Aktionen (Jennings 2001, 39). Obwohl agentenbasierte Systeme mit diesen Mechanismen prinzipiell dazu geeignet sind, die zunehmende Komplexität von Software-Systemen zu beherrschen (Jennings/Sycara/Wooldridge 1998, 7; Jennings 2001), die mit herkömmlichen Technologien nur schwer oder gar nicht handhabbar ist (siehe auch Pokahr/Braubach/Lamersdorf 2005a, 300) oder die über den durch bisherige Vorgehensweisen eingeschränkten Komplexitätsgrad hinausgeht (Krempels et al. 2003, 31), ist wie bei den übrigen Programmiertechniken immer der Anwendungskontext bei der Entscheidung für eine Technologie zu berücksichtigen. Auch wenn Agentensysteme vorteilhaft für verteilte und heterogene Umgebungen eingesetzt werden können (siehe auch Pokahr/Braubach/Lamersdorf

2005a, 300) und es erwartet wird, dass durch ihren Einsatz Software-Architekturen vereinfacht und Entwicklungsprozesse beschleunigt werden können (Kirn 2002, 55), ist der erforderliche Aufwand gegen den Nutzen abzuwägen: Nur in Abhängigkeit vom jeweiligen Kontext können Software-Agenten einen wesentlichen Vorteil gegenüber anderen Programmiertechniken mit sich bringen. Es ist daher im Einzelfall zu prüfen, ob der durch den Einsatz von Agentensystemen induzierte Mehraufwand durch die erwarteten Vorteile gerechtfertigt ist. Erschwerend kommt bei dieser Entscheidung hinzu, dass bisher für die durchgängige Entwicklung von Multiagentensystemen geeignete Werkzeuge fehlen, welche einen nahtlosen Übergang von einer Phase im Vorgehensmodell zur nächsten adäquat unterstützen (Krempels et al. 2003, 32).

Die Darstellung von Agentenplattformen zeigt zwar, dass Standardisierungsbemühungen erkennbar sind. Die Funktionalitäten von Agentenplattformen erreichen aber noch nicht den Umfang derjenigen, die bereits in etablierten, alternativen Technologien für verteilte Systeme gegeben sind. Somit ist für eine breite Akzeptanz von Agentensystemen ihre Integration in bereits etablierte Technologien zu gewährleisten bzw. sind fehlende Funktionalitäten zu spezifizieren und implementieren.

3 Architekturen in der Software-Entwicklung

3.1 Überblick

Ziel von Abschnitt 3 ist die Herausarbeitung derjenigen Eigenschaften von Software-Architekturen, durch die sich letztere zur Erfüllung von flexiblen Anforderungen auszeichnen (Abbildung 3.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1). Dazu werden allgemeine Architekturmuster (Abschnitt 3.2.3) und Architekturen zur Integration von Informationssystemen (Abschnitt 3.2.4) beschrieben. In Abschnitt 3.3 wird dargestellt, wie diese Architekturen auf der Basis eines MAS umgesetzt werden können. Dabei wird insbesondere die Vorteilhaftigkeit von agentenbasierten Systemen herausgearbeitet. Die beschriebenen Architekturen werden in Abschnitt 3.4 analysiert.

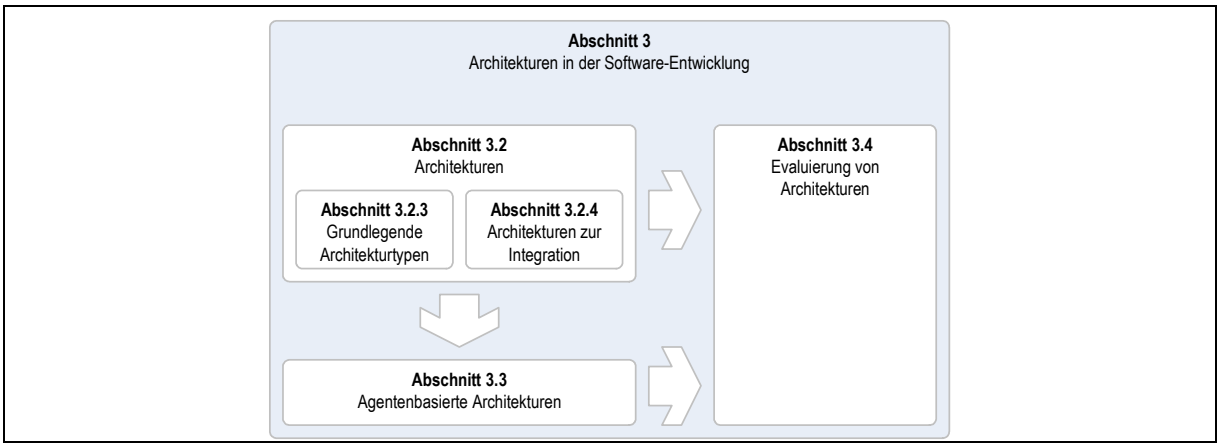

Abbildung 3.1-1: ***Überblick über Abschnitt 3***
Quelle: Eigene Darstellung

3.2 Architekturen

Architekturmuster in der Software-Entwicklung (Perry/Wolf 1992) können nach *Hasselbring* (2006, 51) in Analogie zu Gebäudemustern (Alexander et al. 1977) betrachtet werden. Demnach erscheint es sinnvoll, die Konstruktion eines isolierten Software-Systems mit einem Gebäudebau zu vergleichen. Die Implementierung betrieblicher Informationssysteme und ihre Integration z.B. über den Ansatz der Service orientierten Architektur (siehe Abschnitt 3.2.4.2) gleichen hingegen einem Städtebau, bei dem mehrere Infrastruktureinheiten, also z.B. Gebäude, miteinander in Beziehung gesetzt werden, um ein sinnvolles Ganzes zu erhalten (Hasselbring 2006, 51).

Als Grundlage für Informationssystemarchitekturen werden in den folgenden Abschnitten zunächst die Begriffsbildung einer Software-Architektur, ihre Ziele sowie wesentliche Software-Architekturen und insbesondere diejenigen zur Integration von Informationssystemen beschrieben. Diese Darstellung erlaubt die Begründung der für die vorliegende Arbeit gewählten Architektur.

3.2.1 Begriffsbildung Software-Architektur

Für den systematischen Entwurf von Software-Systemen eignet sich ein Vorgehen entlang einer etablierten Architektur. *Shaw/Garlan* (1996, 1) definieren den Begriff der Software-Architektur als eine Beschreibung von Elementen, aus denen Systeme konstruiert werden, von Interaktionen zwischen diesen Elementen, von Mustern, mit denen die Komposition der Elemente geleitet wird und von Bedingungen zu diesen Mustern. *Hasselbring* (2006, 48) erweitert diese Definition durch den expliziten Bezug der Komponenten zu ihrer Umwelt und durch Prinzipien, die „die Evolution des Systems bestimmen“ (Hasselbring 2006, 48). Mit Software-Architekturen wird eine Reihe von Zielen angestrebt, die im folgenden Abschnitt 3.2.2 beschrieben werden.

3.2.2 Ziele von Software-Architekturen

Als wesentliche Ziele von Software-Architekturen werden bei *Hasselbring* die Aspekte Dokumentation, Kommunikation, Wiederverwendung, Voraussetzung für einen verfeinernden Systementwurf sowie Qualitätssicherung (2006, 50) genannt: Um Software verständlich darzustellen, eignen sich Softwarearchitekturen für die Dokumentation auf hohem Abstraktionsniveau. Neben der Dokumentation werden Architekturen auch zur verbesserten Kommunikation in Entwicklungsteams (Bass/Clements/Kazman 2003, 26) und damit als geeignete Voraussetzung für die spätere Wiederverwendung oder Weiterentwicklung eingesetzt (Hasselbring 2006, 50). Weiterhin tragen Architekturen als erster Schritt zum verfeinernden Systementwurf einen Mehrwert bei (Hasselbring 2006, 50). Da die nachträgliche Anpassung einer Implementierung an die ursprünglichen Anforderungen je später in der Phase der Entwicklung, desto teurer ist, ist eine qualitative Bewertung der Architektur vor der eigentlichen Implementierung sinnvoll (siehe Hasselbring 2006, 50). Auch der Einsatz von etablierten Architekturen unterstützt die Erstellung von qualitativ hochwertigen Software-Systemen. Auf diesem Weg können Software-Architekturen insgesamt zur Steigerung der Qualität der Software, zur Verkürzung von Entwicklungsprozessen sowie zur termingerechten Fertigstellung beitragen. Somit bilden Architekturen einen wesentlichen Bestandteil bei der Entwicklung von Software-Systemen.

Hasselbring (2006, 48-49) nennt eine Reihe von Architekturstilen, die im folgenden Abschnitt 3.2.3 beschrieben werden. Diese Architekturen sind zur Realisierung von Software- bzw. Informationssystemen etabliert.

3.2.3 Grundlegende Architekturtypen

3.2.3.1 Schichtenarchitektur

Schichtenarchitekturen (Dijkstra 1968) sind in der Informatik ein bewährtes (Zweben et al. 1995) und grundlegendes Muster zur Strukturierung von Software-Systemen oder Protokoll-Stacks (vgl. z.B. das ISO-OSI-Modell, Tanenbaum 2003). Ziel dieser Architektur ist die Reduktion der Komplexität durch die Verteilung und Zusammenfassung zusammengehöriger Funktionalitäten auf mehrere Schichten. Die Kommunikation zwischen den Schichten erfolgt über wohl definierte Schnittstellen. So ist es möglich, die unterschiedlichen Schichten mit funktional gleichen auszutauschen. Weiterhin lassen sich damit die Verantwortlichkeiten der

Implementierung und diejenigen der Nutzung einer Schicht trennen, wodurch die Aufgabenteilung in Software-Entwicklungsprojekten unterstützt wird. Die Kommunikation zwischen den Schichten läuft meist zwischen den direkt benachbarten Schichten ab: Aus der Schicht n + 1 (siehe Abbildung 3.2-1) werden Dienste aus der Schicht n aufgerufen. Die von dieser Schicht bereitgestellten Dienste werden damit von der Schicht n + 1 verfeinert. Es kann aber ggf. auch der Aufruf von Diensten über mehrere Schichten hinweg erfolgen (Hasselbring 2006, 48).

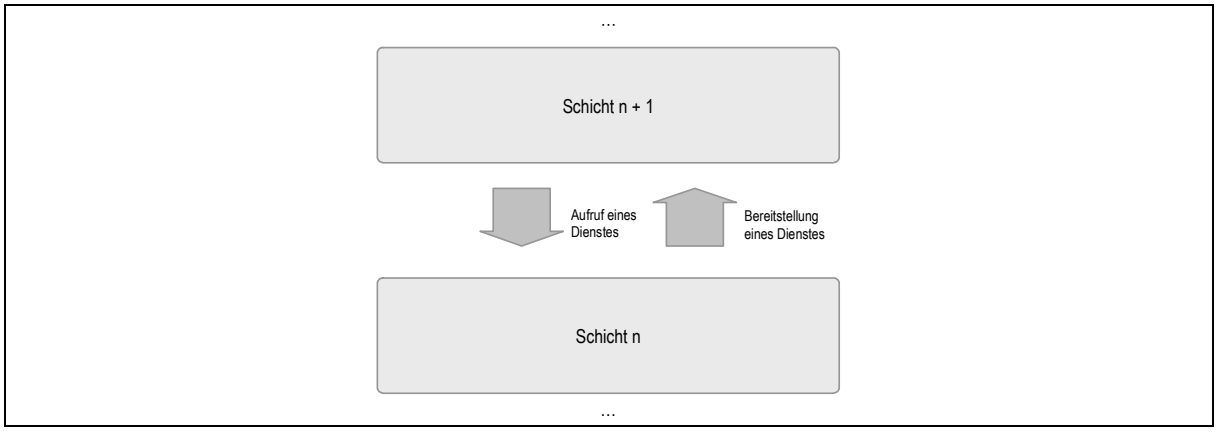

Abbildung 3.2-1: ***Prinzip für eine Schichtenarchitektur***
Quelle: Eigene Darstellung

Eine häufig eingesetzte Konkretisierung der Schichtenarchitektur teilt eine Anwendung in die Bereiche Präsentation, Anwendungslogik und Persistenz auf (DeRemer/Kron 1976). Die genannten Schichten der Präsentation bzw. Anwendungslogik werden oftmals über das MVC-Muster (Model, View, Controller, Krasner/Pope 1988) realisiert. Dabei wird zwischen den Elementen Controller (zur Interaktion mit Benutzereingaben, einzuordnen in die Präsentationsschicht), View (Visualisierung von Daten) und Model (Speicherung der Geschäftslogik) differenziert (siehe auch Abbildung 3.2-2). Diese Architekturmuster finden in der in der vorliegenden Arbeit vorgestellten Agentenlösung Berücksichtigung.

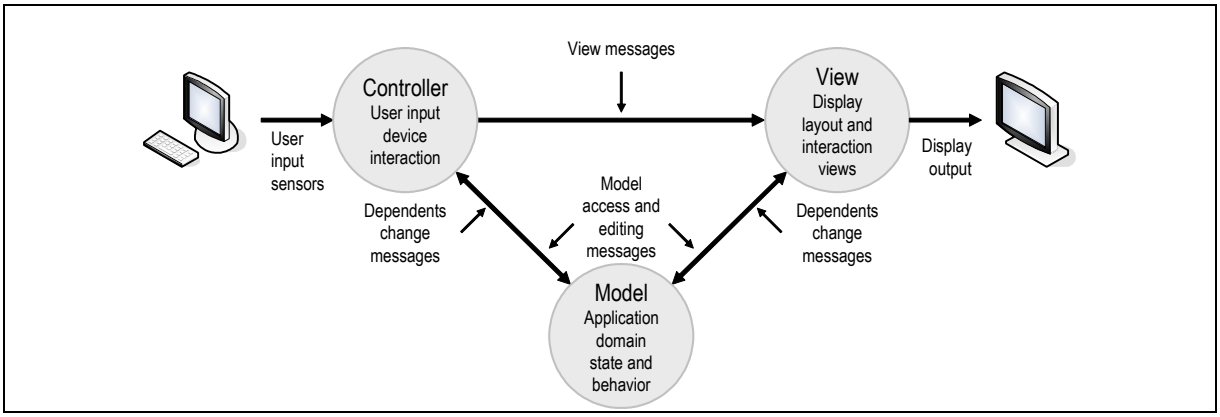

Abbildung 3.2-2: ***MVC-Muster***
Quelle: *Krasner/Pope* (1988, 27)

3.2.3.2 Verteilte Architektur

Verteilte Anwendungen laufen auf lokal verteilten Knoten ab. Um die Kommunikation zwischen diesen Anwendungsteilen zu ermöglichen und von der Netzwerkstruktur zu abstrahieren, wird Middleware eingesetzt. Diese realisiert üblicherweise die folgenden Dienste (Matyas/Maurer 2003, 95):

- *Lebenszyklusdienst*: Erzeugen, Löschen und örtliche Verlagerung von Objekten
- *Namens- und Verzeichnisdienst*: Registrierung von Objekten mit ihrem Namen. Andere Anwendungen können im zentralen Verzeichnisdienst nach Objekten mit bestimmten Namen suchen.
- *Trading-Dienst*: Erweiterung des Namens- und Verzeichnisdienstes zur Suche nach von Objekten angebotenen Funktionalitäten. Es werden Objekte zurückgeliefert, welche eine bestimmte Schnittstelle implementieren.
- *Transaktionsdienst*: Gewährleistung von transaktionssicheren Aktionen in dem verteilten System
- *Persistenzdienst*: Funktionalität für eine Schnittstelle zu einer Datenbank, welche die Daten eines Objektes persistent ablegt

Verteilte Architekturen können in unterschiedlicher Ausprägung umgesetzt werden. In den folgenden Abschnitten 3.2.3.3 mit 3.2.3.5 werden häufig eingesetzte Ausprägungen beschrieben.

3.2.3.3 Client-Server-Architektur

Client-Server-Anwendungen sind typische verteilte Systeme. Diese bestehen aus einer Reihe von Clients, die Anfragen an die Server senden (siehe Abbildung 3.2-3a.). Letztere stellen ihre Dienste zur Verfügung, um die Anfragen der Clients zu bearbeiten. Die Gründe für die weite Verbreitung solcher Systeme können wie folgt zusammengefasst werden (Matyas/Maurer 2003, 95-96):

- *Ortsunabhängigkeit*: Der Aufruf von Diensten entfernter Objekte erfolgt transparent vom eigentlichen und jeweils aktuellen Aufenthaltsort des entfernten Objektes. Somit wird die Flexibilität und Erweiterbarkeit der Anwendung unterstützt.
- *Verbindungsmanagement*: Die eigentliche Kommunikation über das Netzwerk zwischen lokalem und entferntem Objekt wird durch die Middleware geleistet. Das entfernte Objekt kann so verwendet werden, als wenn es lokal vorhanden wäre.
- *Skalierbarkeit*: Die Objekte können transparent auf andere Knoten migriert werden.
- *Performanz*: Objekte werden denjenigen Knoten zugeordnet, auf denen umfangreiche Operationen durchgeführt werden können. Somit können Kapazitäten auf Knoten ohne umfangreiche Ressourcen eingespart werden.
- *Sicherheit*: Die Middleware übernimmt Sicherheitsdienste, indem Rechte zum Ausführen bestimmter Dienste von Objekten hinterlegt werden und ihre Durchsetzung gewährleistet wird.

- *Lastverteilung*: Um einen performanten Betrieb zu gewährleisten, wird die Last auf verschiedene Knoten verteilt.
- *Fehlertoleranz*: Dienste können transparent bei Ausfällen eines Serversystems auf andere Knoten migiert werden.
- *Plattformunabhängigkeit*: Die zugrunde liegenden Kommunikationsprotokolle sind plattformunabhängig definiert. Somit können heterogene Systeme die Dienste der verteilten Anwendung in Anspruch nehmen.
- *Integration von Altanwendungen*: Die Integration von Altanwendungen kann durch den Einsatz von Middleware erleichtert werden.

Da die Vorteile von Client-Server-Anwendungen so umfangreich sind, konnten sich letztere hinsichtlich ihrer Einsatzhäufigkeit weit verbreiten. Dennoch kann auch eine Reihe von Defiziten von Client-Server-Applikationen identifiziert werden: Die steigenden Anforderungen bezüglich des Anwendungsspektrums, aber auch der zunehmenden Teilnehmerzahlen und des Datenaufkommens erschweren die Konstruktion von verteilten Anwendungen auf der Basis des Client-Server-Prinzips (Steinmetz/Wehrle 2004, 51). Den neuen Anforderungen der Skalierbarkeit, Sicherheit und Verfügbarkeit sowie der Flexibilität zur Integration neuer Dienste und der Dienstgüte kann die Form der Client-Server-Architektur jedoch nicht gerecht werden (Steinmetz/Wehrle 2004, 51). Wesentliche Herausforderungen sind dabei die zentralen Einheiten, die durch ihre Eigenschaft als Flaschenhals oder Angriffspunkt Probleme verursachen können (Steinmetz/Wehrle 2004, 51). Um die genannten Anforderungen erfüllen und Problemstellungen bewältigen zu können, bieten Peer-to-Peer-Architekturen wesentliche Lösungsmöglichkeiten. Diese werden im folgenden Abschnitt 3.2.3.4 beschrieben.

3.2.3.4 Peer-to-Peer-Architektur

Peer-to-Peer-Architekturen können als Erweiterung der Client-Server-Architektur und auch als adäquates Mittel betrachtet werden, um die in Abschnitt 3.2.3.3 genannten Probleme von Client-Server-Architekturen zu lösen (Steinmetz/Wehrle 2004, 51). *Steinmetz/Wehrle* (2004) definieren Peer-to-Peer-Netzwerke wie folgt:

> Ein Peer-to-Peer-Netzwerk kann als ein „System mit vollständig dezentraler *Selbstorganisation* [Herv. durch Verf.] und *Ressourcennutzung* [Herv. durch Verf.]“ (Steinmetz/Wehrle 2004, 52) verstanden werden.

Die wesentlichen Eigenschaften sowie die Strukturierung von Peer-to-Peer-Architekturen werden im Folgenden detailliert beschrieben:

Die Selbstorganisation kann in die folgenden Teilbereiche strukturiert werden (Steinmetz/Wehrle 2004, 52-53):

- Zur Nutzung der angebotenen Dienste interagieren die Peers autonom ohne weitere zentrale Dienste miteinander. Die Koordination bei zentralen Client-Server-Diensten wird damit durch die Kooperation zwischen den Peers zugunsten der Vermeidung von Flaschenhälsen und der Gewinnung der Zuverlässigkeit aufgegeben.

- Für den Zugriff auf und den Transfer von Ressourcen interagieren die Peers direkt ohne zentrale Instanzen.
- Peers nehmen sowohl die Rolle eines Clients als auch eines Servers ein. Damit kann die Flexibilität der Dienstbereitstellung erlangt werden.
- Peers sind gleichberechtigt und autonom hinsichtlich der Nutzung der Ressourcen.
- Ressourcen sollen möglichst ohne zentrale Dienste lokalisiert werden können.

Die Eigenschaft der dezentralen Ressourcennutzung kann wie folgt untergliedert werden (Steinmetz/Wehrle 2004, 52):

- Die auf den Peers vorhandenen Betriebsmittel Bandbreite, Speicherplatz und Rechenleistung werden gleichmäßig und verteilt genutzt.
- Die Peers nutzen die untereinander zur Verfügung gestellten Betriebsmittel.
- Peers sind über ein Netzwerk miteinander verknüpft und meist verteilt.
- Die variable Konnektivität erlaubt den Wechsel der Internet-Adresse eines Peers. Damit stehen Daten nicht mehr bezogen auf Adressen, sondern in Bezug auf Inhalte zur Verfügung. Die Wegewahl erfolgt deshalb auch inhaltsorientiert.

Wesentliche Eigenschaften eines Peer-to-Peer-Systems im Vergleich zu Client-Server-Ansätzen sind also der Übergang von der Koordination zur Kooperation, von zentralen zu dezentralen Komponenten sowie von der Kontrolle zu Anreizen (Steinmetz/Wehrle 2004, 53). Ein Ziel bei Peer-to-Peer-Systemen ist die Etablierung eines Gleichgewichtes zwischen der Ressourcennutzung zweier Peers (Steinmetz/Wehrle 2004, 53). Auf diesem Weg wird der Forderung nach einer Umsetzung des genannten Anreizprinzips genügt.

Hinsichtlich der chronologischen Entwicklung von Peer-to-Peer-Netzwerken können nach *Steinmetz/Wehrle* die zwei Ausprägungen *unstrukturierte* und *strukturierte* Systeme differenziert werden (2004, 53-54), welche im Folgenden, angelehnt an die Ausführungen bei *Steinmetz/Wehrle*, beschrieben werden:

Bei unstrukturierten Verfahren erfolgt die Adressvermittlung über einen zentralen Server. Dabei wird auch von einer hybriden Struktur gesprochen (siehe Abbildung 3.2-3b.). Alternativ können die Ressourcen über den Mechanismus des Flutens lokalisiert werden, bei dem an die Teilnehmer des Peer-to-Peer-Netzes Suchnachrichten versendet werden. Nachteilig dabei ist jedoch, dass die beiden genannten Varianten nicht die Anforderung der Skalierbarkeit erfüllen. Der Server kann zum Flaschenhals werden und die Flutung verursacht beträchtliche Netzlast. Die Suche nach einer Ressource nimmt dabei die Komplexität von mindestens O(N) ein (Steinmetz/Wehrle 2004, 54).

Der strukturierte Ansatz (siehe Abbildung 3.2-3c.) verfolgt die Ziele der Skalierbarkeit, Verfügbarkeit und Fehlertoleranz (Steinmetz/Wehrle 2004, 54) und strebt damit die Beseitigung der genannten Nachteile von unstrukturierten Verfahren an. Die Speicherung von Daten erfolgt dabei auf der Basis von verteilten Indexstrukturen (Hash-Tabellen), um Inhalte adressieren zu können. In einer verteilten Hash-Tabelle erhält jeder beteiligte Knoten einen Teil des Suchraumes. Für die Gewährleistung der Redundanz werden diese Daten auch in benachbar-

ten Knoten repliziert vorgehalten. Die Hash-Tabelle eines Knotens erhält O(log N) Einträge (Steinmetz/Wehrle 2004, 54). Mit diesen Informationen kann von einem Einstiegsknoten eine Wegesuche zur angefragten Ressource erfolgen. Die Suche in einer solchen Struktur ist hinsichtlich ihrer Komplexität leistungsfähig, da eine Komplexität von O(log N) erzielt werden kann (Steinmetz/Wehrle 2004, 54).

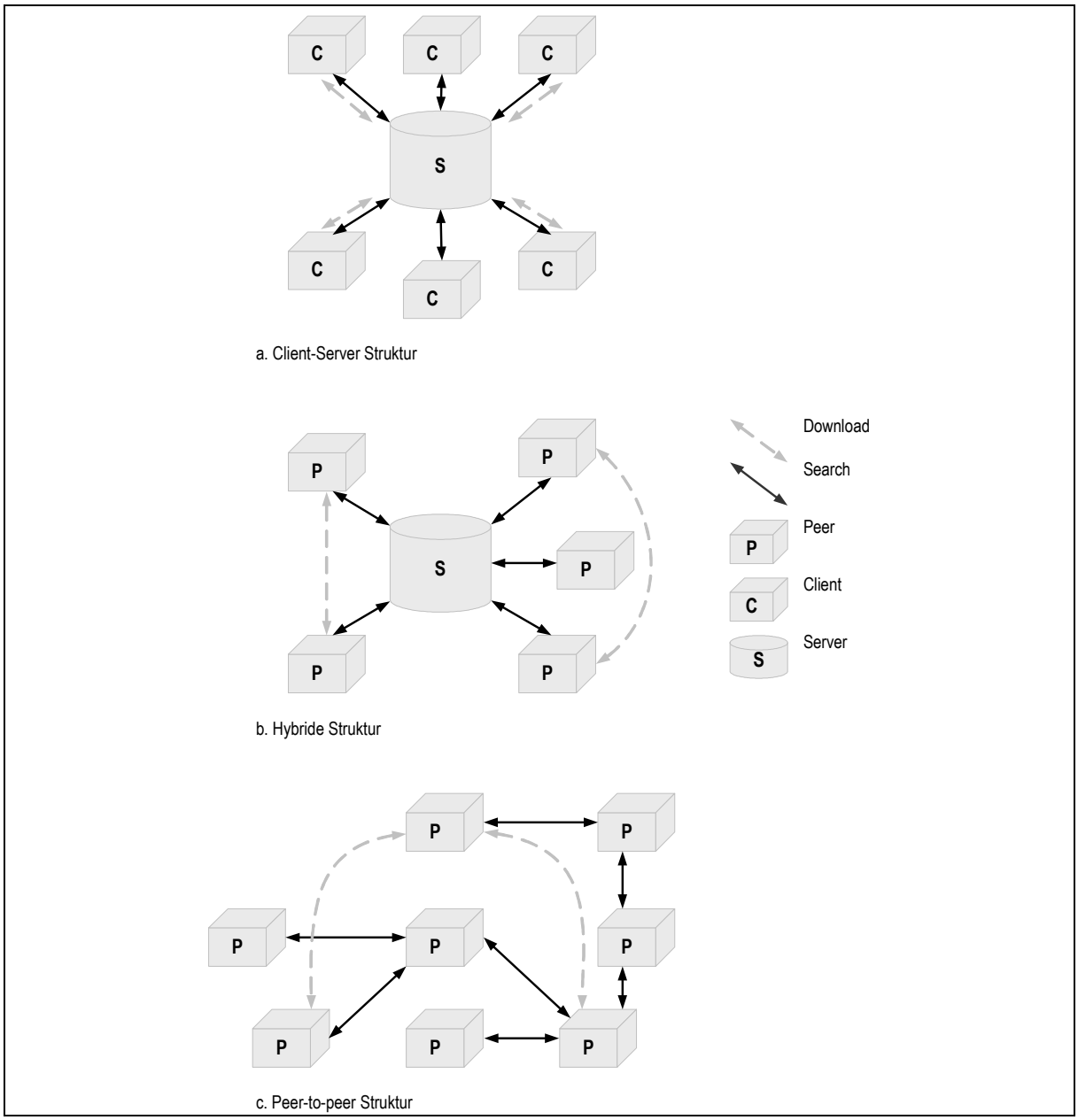

Abbildung 3.2-3: ***Darstellung von Client-Server-, hybriden und Peer-to-Peer-Strukturen***
Quelle: *Steinmetz/Wehrle* (2004, 53)

Die verteilten Hash-Tabellen besitzen zudem Verwaltungsmöglichkeiten der verteilten Datenstruktur, Verfahren zur redundanten Datenhaltung sowie Mechanismen zum Auffinden von Knoten mit möglichst geringer Weglänge (Steinmetz/Wehrle 2004, 54). Die Verwaltungsoperationen wie die Integration neuer Inhalte oder Knoten sowie die Behandlung von Fehlerfällen besitzen nach *Steinmetz/Wehrle* (2004, 54) die Komplexität O(log N) bzw. O(log² N).

Anhand dieser Ausführungen wird deutlich, dass Peer-to-Peer basierte Anwendungen beim Anwachsen sowohl hinsichtlich Netzgröße als auch bezüglich Anwenderzahl skalieren können. Somit können die Performanz von Peer-to-Peer-Architekturen gegenüber herkömmlichen Client-Server-Architekturen gesteigert und insbesondere neue Anforderungen an Software-Architekturen erfüllt werden.

3.2.3.5 Komponentenorientierte Architektur

Im Folgenden werden, ausgehend von dem Bedarf für komponentenorientierte Software-Entwicklung, die Eigenschaften eines solchen Systems sowie diejenigen der einzelnen Komponenten beschrieben. Ergänzt wird diese Darstellung durch einen Ansatz zur formalen Spezifikation eines Systems aus mehreren Komponenten. Diese Ausführungen dienen als Grundlage für die Beschreibung von ausgewählten Implementierungsmöglichkeiten für ein komponentenorientiertes System in den Anhängen A.1mit A.3.

Szyperski (2002, 4) teilt Software-Entwicklung in zwei unterschiedliche Lager: Kundenspezifische und Standardlösungen. Obwohl erstere meist die Anforderungen besser erfüllen können, ist ihre Entwicklung oftmals kostenintensiv (Szyperski 2002, 5). Standardlösungen werden als solche ausgeliefert, womit auch die Entwicklung kostengünstiger gestaltet werden kann. Jedoch können Standardlösungen auch umfangreiche Änderungen in den Geschäftsprozessen verursachen (Szyperski 2002, 5).

Die Komponentenorientierung kann die Lücke zwischen diesen beiden Extremen schließen (Szyperski 2002, 6), indem Standardkomponenten verwendet werden, die zu einem Gesamtsystem zusammengestellt werden. Software-Komponenten erlauben die Komposition von Lösungen aus unterschiedlichen, meist vorgefertigten Komponenten und ermöglichen damit eine kostengünstige und flexible Konstruktion einer Software-Lösung, weil die Komponenten entsprechend den gegebenen Anforderungen ausgewählt oder ggf. angepasst werden können (Szyperski 2002, 6). Somit besteht nicht die Notwendigkeit der Anpassung der Geschäftsprozesse an die Software.

Ein komponentenorientiertes System besteht aus einer Menge von Komponenten, die zu einem System zusammengestellt werden können (Szyperski 2002, 4). Damit erlaubt die komponentenorientierte Software die Wiederverwendung von Software-Elementen auf höherem Abstraktionsniveau, als sie z.B. bei der Objektorientierung gegeben ist (Szyperski 2002, 3).

Software-Komponenten sind durch ihre Unabhängigkeit gekennzeichnet und als ausführbare Einheit ausgelegt, womit die Eigenschaften der Unabhängigkeit vom Hersteller und dem Deployment sowie die robuste Integration ermöglicht werden (Szyperski 2002, 4). Wichtig für die Betrachtung der Komponentenorientierung ist die Differenzierung zwischen dem Aspekt eines Bauplanes für eine Komponente und ihrer Instanzbildung (Szyperski 2002, 8-10). Komponenten besitzen demnach nach *Szyperski* (2002, 36-37) folgende Eigenschaften:

- *Einheit mit unabhängigem Deployment* (Szyperski 2002, 36): In einer Komponente werden die Funktionalitäten gekapselt, um von der Umgebung bzw. von anderen Komponenten unabhängig zu sein. Eine Komponente wird ausschließlich in ihrer Gesamtheit eingesetzt.

- *Einheit zur Komposition durch Dritte* (Szyperski 2002, 36): Komponenten sind in sich abgeschlossen. Dritte besitzen keine Kenntnis über die Implementierungsdetails einer Komponente, weil diese gekapselt werden. Die Schnittstellen zur Verwendung einer Komponente müssen wohl definiert sein, um die benötigten und angebotenen Dienste zu spezifizieren. Der Aufruf dieser Dienste erfolgt meist über einen zentralen Bus (Hasselbring 2006, 49). Im Gegensatz zu Service orientierten Architekturen (siehe Abschnitt 3.2.4.2) wird in der Komponentenorientierung insbesondere der Installationskontext berücksichtigt (Hasselbring 2006, 49). Diese Kontextabhängigkeiten definieren den Kontext der Komposition und des Deployments (Szyperski 2002, 44): Der Kompositionskontext spezifiziert die Regeln für die Komposition der Komponente. Der Deployment-Kontext gibt die Regeln für das Deployment, die Installation und die Aktivierung einer Komponente an. Somit ist in der Komponentenorientierung die Umgebung zu spezifizieren, für die die Komponente implementiert ist. Dabei kann nach *Szyperski* (2002, 44) zwischen den konkreten Umgebungen Java (für eine überblicksartige Darstellung siehe Anhang A.1), CORBA (siehe Anhang A.2) und Microsoft COM bzw. CLR (siehe Anhang A.3) differenziert werden.
- *Einheit ohne nach außen sichtbaren Zustand* (Szyperski 2002, 36): Mit einer Black-Box-Sicht ist bei einer Komponente ihr Zustand von außen nicht erkennbar.

Die Definition für eine Komponente kann damit wie folgt zusammengefasst werden (Szyperski 2002, 41):

> **„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".**

Die formale Spezifikation eines Systems aus Komponenten kann z.B. mit FOCUS (Broy/Stølen 2001) vorgenommen werden. FOCUS unterstützt die Spezifikation, die Verfeinerung und die Verifikation für interaktive Systeme (Broy/Stølen 2001, 8). Dazu werden Werkzeuge zur Verfügung gestellt, die eine mathematische und logische Fundierung besitzen (Broy/Stølen 2001, 8) und in unterschiedlichen Phasen eines Systementwicklungsprozesses eingesetzt werden können.

Solche interaktiven Systeme sind in eine Umgebung eingebettet und interagieren mit ihr. Mit der mathematisch fundierten Notation FOCUS werden Systeme und ihre Komponenten mit unterschiedlichen Ausdrucksmöglichkeiten formal beschrieben. Komponenten können dabei über eine hierarchische Struktur in weitere Komponenten zerlegt werden (Broy/Stølen 2001, 2). Differenziert wird zwischen einer Black-Box- und einer White-Box-Sicht auf Komponenten. Die Kommunikation zwischen Komponenten erfolgt über typisierte Kanäle. Statische Eigenschaften beschreiben nicht veränderliche Charakteristika (Broy/Stølen 2001, 2) wie z.B. die Schnittstellen von Komponenten. Um das dynamische Verhalten der Komponenten zu beschreiben, werden Relationen zwischen den Ein- und Ausgabekanälen angegeben. Ein- und Ausgabe werden als ein Strom von Datenelementen betrachtet. Für die Spezifikation können unterschiedliche Varianten, abhängig von der Phase im Entwicklungsprozess, gewählt werden. Insbesondere wird zur Reduktion der Komplexität der formalen Spezifikationstechniken auch eine grafische Notation beschrieben, die sich an etablierten Beschreibungstechniken anlehnt und erweitert (Krüger/Prenninger/Sandner 2004) bzw. auf solche übertragen (Krüger

2000) werden kann. Diese Spezifikationsstile sind jedoch alle zueinander semantisch äquivalent.

In FOCUS ist der Kompositionskontext im Sinne der Komponentenorientierung durch die Bestimmung der Ein- und Ausgaberelationen gegeben. Durch letztere wird determiniert, wie eine Komponente mit anderen Komponenten komponiert werden kann. Da der Deployment-Kontext abhängig von der verwendeten Technologie ist, wird bei FOCUS keine geeignete Spezifikationstechnik angegeben.

Die Ausführungen zur Komponentenorientierung zeigen, dass auf der Basis von Komponenten eine Wiederverwendung auf höherer Abstraktionsstufe möglich ist, als sie in der Objektorientierung gegeben ist. Zur Realisierung eines komponentenorientierten Systems existieren mit Java, CORBA und COM bzw. CLR etablierte Technologien.

3.2.4 Architekturen zur Integration

Abschnitt 3.2.3 fokussierte grundlegende Architekturtypen. Im Hinblick auf die Zielvorgabe der vorliegenden Arbeit zur Erarbeitung eines Lösungsvorschlags für eine Integrationslösung im inhärent verteilten Gesundheitswesen werden in den folgenden Abschnitten 3.2.4.1 mit 3.2.4.4 Architekturen zur Integration beleuchtet.

3.2.4.1 Enterprise Application Integration

Ausgehend von einer Definition der EAI werden in diesem Abschnitt die Ziele, grundlegende Architektur, Integrationsebenen, -aufgaben und -umsetzungen einer EAI überblicksartig beschrieben.

Strüver (2006, 13) definiert den Begriff der Enterprise Application Integration (EAI) wie folgt:

> **„EAI als Begriff subsumiert Integrationstechnologien im Allgemeinen sowie eine bestimmte Art und Weise, diese anzuwenden. Die Ausgangslage für EAI ist die einer heterogenen, über Jahre gewachsenen IT-Landschaft, in der ein ausreichend hoher Bedarf für [eine prozessorientierte, Anm. des Verf.] Integration herrscht, [Interpunktion durch den Verf. eingefügt] um Integration an sich als Problem auftreten zu lassen. EAI stellt einen möglichen Lösungsansatz für dieses Problem dar".**

Nach dieser Definition wird unter EAI nicht nur eine Familie von Integrationstechnologien (Strüver 2006, 13), sondern insbesondere eine Beschreibung des Vorgehens bei der EAI verstanden (Strüver 2006, 17). Die wesentlichen Ziele einer EAI hinsichtlich eingesetzter Technologien ist die Redundanzfreiheit (Strüver 2006, 20) und bezüglich des Vorgehens die „Flexibilisierung [..] bei der Veränderung der Prozesse" (Strüver 2006, 20).

EAI basiert auf einem Enterprise Software Bus-Modell (Krafzig/Banke/Slama 2006, 24), um die Systeme miteinander zu verbinden. Dieses Prinzip für eine EAI-Architektur ist in Abbildung 3.2-4 dargestellt. Dabei wird ersichtlich, dass die Anwendungen untereinander meist nur indirekt über den Versand von Nachrichten über den Message Broker interagieren (Kossmann/Leymann 2004, 119). Neben der in Abbildung 3.2-4 dargestellten Bus Topologie

können weitere Formen von Integrationstopologien differenziert werden, die auch oftmals zu Mischformen komponiert werden (Strüver 2006, 32-36):

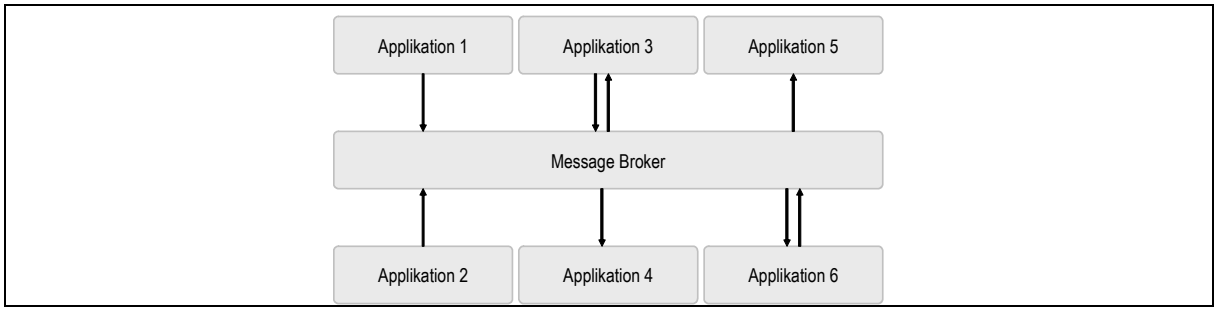

Abbildung 3.2-4: ***EAI-Architekturprinzip***
Quelle: *Kossmann/Leymann* (2004, 119)

- *Point-to-Point Topologie*: Jedes Informationssystem ist mit jedem anderen Informationssystem direkt verbunden.
- *Hub-and-Spoke Topologie*: Gewährleistung der Verbindung zwischen den Informationssystemen durch ein zentrales Element sowie der Integrationslogik auf allen Ebenen der EAI

Auf der Architekturebene kann EAI, ähnlich dem Vergleich der Architektur mit Software-Systemen (Gebäudemuster als Metapher für Software-Architekturen, siehe Abschnitt 3.1), als Analogie zum Städtebau (als Metapher für integrierte Informationssysteme) betrachtet werden (Hasselbring 2006, 51). Dies hängt damit zusammen, dass bei der EAI eine unterschiedliche Anzahl heterogener Informationssysteme mit jeweils anderen Aufgaben zu einem ganzheitlichen betrieblichen Informationssystem integriert werden soll. Beim Städtebau sind unterschiedliche Interessen, wie z.B. eine möglichst ausgeprägte Infrastruktur und ein angenehmer sowie ruhiger Wohnraum, zu berücksichtigen (Hasselbring 2006, 51). Analog dazu sind auch bei der EAI unterschiedliche Aufgaben der jeweiligen Teilsysteme zu integrieren. Dazu ist z.B. ein Kompromiss zwischen einer möglichst geringen Anzahl eingesetzter EAI-Werkzeuge und einer umfassenden Integration aller beteiligten Systeme auf allen Ebenen zu schließen.

Der Fokus der EAI liegt auf der Integration von Informationssystemen (Keller 2002; Kossmann/Leymann 2004, 117) meist innerhalb einer Organisation (Strüver 2006, 13 und 25), während die Integration über Organisationsgrenzen hinweg oftmals in den Hintergrund rückt (Krafzig/Banke/Slama 2006, 24). Mit der EAI sollen weiterhin die Aspekte Daten, Objekte und Prozesse (Strüver 2006, 29.30) integriert werden, wobei die Integrationsrichtung durch Geschäftsprozesse auf operativer Ebene gegeben ist (Strüver 2006, 13).

Die logischen Aufgaben der zur Integration eingesetzten Technologien und damit des Message Brokers können mit dem Transport der Daten über eine Middleware (Strüver 2006, 36-37), der Transformation der Daten für den Empfänger durch eine Transformationskomponente (Strüver 2006, 37-38), dem Routing oder der Weiterleitung der Daten an den Empfänger (Strüver 2006, 38) sowie der Prozesslogik (Strüver 2006, 38-39) „als Mittel zur Steuerung der Integrationslogik auf der Objekt- und Datenebene“ (Strüver 2006, 39) zusammengefasst werden. Die konkrete Realisierung dieser logischen Aufgaben kann, jeweils differenziert über

die Integrationsaspekte, nach *Strüver* beispielhaft mit den im Folgenden genannten Technologien durchgeführt werden (2006, 40-62):

- *Konnektivität*: Gewährleistung der Verbindung zwischen Informationssystemen mit Ansätzen wie RPC (Remote Procedure Call), Java RMI™ (Remote Method Invocation), HTTP (Hypertext Transfer Protocol) oder Web Services; Transformation von unterschiedlichen Datenformaten durch Adapter
- *Datenintegration*: Integration unterschiedlicher Datenbankmanagementsysteme mit Ansätzen wie JDBC™ (Java Database Connectivity) oder ODBC (Open Database Connectivity)
- *Objektintegration*: Kommunikation zwischen Software-Objekten oder Komponenten mit Technologien wie CORBA, Enterprise JavaBeans™ (EJB™), Component Object Model (COM) oder SAP® Business Application Programming Interface (BAPI®)
- *Prozessintegration*: „Integration der Ablauflogik“ (Strüver 2006, 57) über die Alternativen „Ablauflogik direkt im Programmiercode“ (Strüver 2006, 58) oder vollständige „Entkopplung der Prozesslogik vom Programmiercode“ (Strüver 2006, 58)

Da in der vorliegenden Arbeit die Betrachtung von Integrationstechnologien fokussiert wird, wird auf die Darstellung eines möglichen EAI-Vorgehens im Folgenden verzichtet und dafür auf die Ausführungen bei *Strüver* (2006, 227-251) verwiesen.

Mit den Aussagen in diesem Abschnitt kann eine EAI zusammenfassend als ein Treiber für den Einsatz einer Service orientierten Architektur betrachtet werden (Krafzig/Banke/Slama 2006, 215), um die Integration auf einer höheren Abstraktionsebene als bei der EAI auf der Basis von Diensten zu realisieren. Deshalb werden die grundlegenden Prinzipien zu Service orientierten Architekturen im folgenden Abschnitt dargestellt.

3.2.4.2 Service orientierte Architektur

Ausgehend von der Notenwendigkeit der Einführung von Service orientierten Architekturen (SOA) werden in den folgenden Ausführungen Ziele und Elemente solcher Architekturen beschrieben. Diese Elemente werden in typische Architekturebenen einer SOA strukturiert. Abgeschlossen wird dieser Abschnitt durch einen kurzen Überblick über mögliche Realisierungstechnologien für eine SOA.

Die in den Abschnitten 2.2.2 mit 2.2.5 beschriebenen Programmiertechniken können auch aus der Perspektive eines Dienstes betrachtet werden. Die Berechungsvorschriften der applikativen, prozeduralen oder objektorientierten Programmierung können jeweils als ein Dienst aufgefasst werden, der von Aufrufern benutzt werden kann, um die angebotenen Aufgaben erfüllt zu bekommen (siehe dazu auch Krafzig/Banke/Slama 2006, 16). Diese Programmiertechniken sind jedoch hinsichtlich ihres Abstraktions- und Granularitätsgrades der Funktionalitäten, die den Aufrufern der Schnittstellen zur Verfügung gestellt werden, zu fein granular, um eine effiziente Wiederverwendung oder Verteilung zu unterstützen (Krafzig/Banke/Slama 2006, 18). So werden mit der Objektorientierung aus dieser Perspektive meist monolithische Systeme entwickelt (Szyperski 2002, 11). Über die Komponentenorientierung (Abschnitt 3.2.3.5) wird ein weiterer Schritt zur Wiederverwendung von Elementen auf höherem Abstraktionsniveau eingeführt. Dieser Trend zur Abstraktion wird in Service orientierten Architekturen fortgesetzt; denn diese erlauben im Gegensatz zur Objektorientierung den Einsatz von grob granularen Komponenten mit einfachen Zugriffsmustern (Krafzig/Banke/Slama 2006, 25), der

für den Aufruf entfernter Dienste geeignet ist (Krafzig/Banke/Slama 2006, 18). Dabei müssen sich Dienstleister und Benutzer dieses Dienstes nicht notwendigerweise auf dem gleichen Knoten befinden, sondern können auch lokal verteilt sein, womit der Begriff des Dienstes wesentlich bestimmt wird (Krafzig/Banke/Slama 2006, 19). Die genannten Eigenschaften sind insbesondere für die Integration auf der Ebene der Geschäftslogik erforderlich (siehe Krafzig/Banke/Slama 2006, 18-19).

Eine SOA zielt auf die Verbesserung der Agilität der Informationssysteme in einem Unternehmen ab (Krafzig/Banke/Slama 2006, 239). Dabei wird diesen Unternehmen die Möglichkeit eingeräumt, ihre Informationssysteme flexibel an Veränderungen in funktionalen Anforderungen oder der Geschäftsstrategie anzupassen, ohne jedoch die Einschränkung auf eine einzige Middleware oder EAI-Plattform hinnehmen zu müssen (Krafzig/Banke/Slama 2006, 24). Dazu sollen in einer Service orientierten Architektur die Geschäftslogik und Daten von Unternehmen, die in Applikationen, Datenbanken und Altsystemen liegen, wirksam eingesetzt werden (Krafzig/Banke/Slama 2006, 24). Um diesen Anforderungen der Unterstützung von Geschäftsprozessen in kürzer werdenden Zyklen sowie der flexiblen Reaktion auf Änderungen in der Unternehmensstruktur geeignet begegnen zu können, stellt die Service orientierte Architektur ein adäquates Instrument dar (Richter/Haller/Schrey 2005, 413) und wird wie folgt definiert:

Eine SOA ist nach *Krafzig* (2006, 57) eine Software-Architektur, die aus den Teilen Applikations-Frontend, Dienst, Dienstverzeichnis und Dienstbus besteht.

Durch die Komponenten Applikations-Frontend, Dienst, Dienstverzeichnis und Dienstbus einer SOA (siehe Abbildung 3.2-5) werden die im Folgenden beschriebenen Aufgaben wahrgenommen (siehe für die Ausführungen im Weiteren Krafzig/Banke/Slama 2006, 56-66):

Durch das Applikations-Frontend wird das gesamte Informationssystem gesteuert. Dabei kann ersteres ggf. eine grafische Benutzeroberfläche besitzen. Dienste werden durch das Applikations-Frontend aufgerufen, wodurch die Steuerung des Informationssystems erfolgt.

Ein Dienst einer SOA ist in einer Implementierung umgesetzt, welche die Geschäftslogik und Daten kapselt. Dienste bieten somit Geschäftsfunktionalität an und bündeln diese Funktionalitäten zusammen mit den jeweils erforderlichen Daten. Die Dienste werden in einer SOA direkt auf Geschäftseinheiten abgebildet, womit SOA eine Integration auf Geschäftsebene und weniger auf technischer Ebene erlaubt (Krafzig/Banke/Slama 2006, 24). Die Dienste kapseln damit jeweils modulorientiert vertikale Funktionalität für bestimmte Geschäftsvorgänge (siehe auch Krafzig/Banke/Slama 2006, 213). Dienste sind deshalb leichtgewichtig und rufen zur Abarbeitung von Anfragen ggf. Funktionalitäten anderer Dienste auf. Durch diese Anwendung des Schichtenprinzips wird die Komplexität der Anwendung reduziert. Weiterhin besitzt ein Dienst einen Contract, durch den die Funktionalität, Benutzung und Bedingungen für den Dienstenutzer formal, z.B. durch WSDL (siehe dazu auch Abschnitt 3.2.4.4), spezifiziert werden. Durch das Interface wird der angebotene Dienst physikalisch sichtbar. *Krafzig/Banke/Slama* (2006, 69-82) differenzieren entsprechend der Funktionalität von Diensten die folgenden Diensttypen (Krafzig/Banke/Slama 2006, 69):

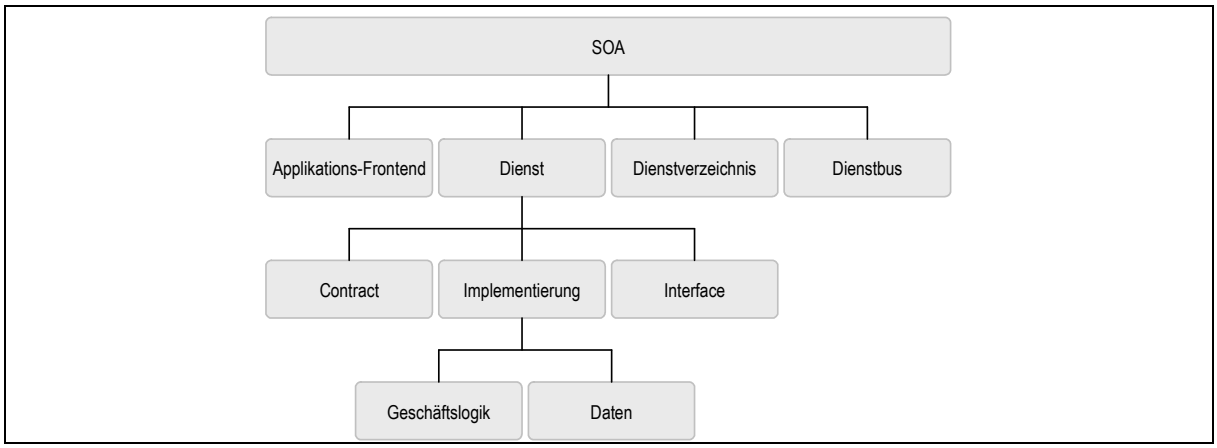

Abbildung 3.2-5: ***Elemente einer Service orientierten Architektur und ihre Relationen***
Quelle: In Anlehnung an *Krafzig/Banke/Slama* (2006, 57)

- Bei *basalen Diensten* (Krafzig/Banke/Slama 2006, 70-73) werden *datenzentrierte* (Krafzig/Banke/Slama 2006, 70-72) und *logikzentrierte Dienste* (Krafzig/Banke/Slama 2006, 72-73) unterschieden: Durch erstere wird das Management von persistenten Daten einschließlich Speicherung und Zugriff auf Daten, Sperrmechanismen und Transaktions-Management realisiert. Im Vergleich zu einer Schichtenarchitektur (siehe Abschnitt 3.2.3.1) wird der datenzentrierte Dienst in Abhängigkeit von Geschäftsobjekten in vertikaler Richtung implementiert. Logikzentrierte Dienste kapseln Algorithmen für umfangreiche Berechnungen oder Regeln aus der jeweiligen Domäne.
- *Intermediäre Dienste* können in *Technology Gateways* (Krafzig/Banke/Slama 2006, 74-75), *Adapter* (Krafzig/Banke/Slama 2006, 75), *Façades* (Krafzig/Banke/Slama 2006, 75-78) sowie *Functionality-Adding-Dienste* (Krafzig/Banke/Slama 2006, 78) klassifiziert werden: Die Aufgabe von Technology Gateways ist die Transformation zwischen unterschiedlichen Technologien. Werden bspw. zwischen Applikations-Frontend und einem basalen Dienst unterschiedliche Technologien eingesetzt, agiert das Technology-Gateway beim Applikations-Frontend als Proxy-Objekt und mediiert die Kommunikation zum basalen Dienst. Ein Adapter ist ein Spezialfall eines Technology Gateways und bildet die Signaturen und Nachrichtenformate eines Dienstes auf die Anforderungen eines Dienstnutzers ab. Façades realisieren unterschiedliche Sichten auf einen oder mehrere Dienste und nehmen deshalb oft die Rolle eines Technology Gateways und bzw. oder Adapters ein. Dabei werden je nach Anforderung Dienste ausgeblendet oder aggregiert. Somit wird insgesamt die Komplexität reduziert. Functionality-Adding-Dienste erweitern einen Dienst um weitere Funktionalität, indem erstere als eigenständige Dienste realisiert werden, die den ursprünglichen Dienst ergänzen, ohne ihn jedoch zu verändern.
- In *prozesszentrierten Diensten* werden Geschäftsprozesse gekapselt.
- *Öffentliche Unternehmensdienste* dienen dazu, Funktionalitäten über Unternehmensgrenzen hinaus anderen Geschäftspartnern zur Verfügung zu stellen.

Bei der bisherigen Beschreibung einer SOA entsprechend den Ausführungen bei *Krafzig/Banke/Slama* (2006, 55-66) werden die zu integrierenden Systeme nicht explizit erwähnt. Diese sind aber implizit durch die Kapselung der Daten und Geschäftslogik dieser Applikationen in Diensten als Elemente einer SOA gegeben.

Aus den bisherigen Ausführungen wird ersichtlich, dass ein Dienst ein wesentliches Element in einer Service orientierten Architektur darstellt. Deshalb wird der Dienstbegriff, zusammengefasst aus den obigen Darstellungen, wie folgt definiert:

Ein Dienst im Sinne einer SOA stellt über eine wohl definierte Schnittstelle Dienstnutzern Funktionalität zur Erbringung der spezifizierten Dienstleistungen zur Verfügung. Dabei werden alle zur Diensterbringung erforderlichen Funktionalitäten entsprechend dem Geheimnisprinzip implementiert. Das Dienstspektrum umfasst einen vertikalen Umfang, der über eine Geschäftseinheit definiert ist, für die eine Dienstleistung bereitgestellt wird.

In einer Service orientierten Architektur werden Dienste über einen Bus lose gekoppelt (Hasselbring 2006, 49). Durch den Dienstbus werden die Dienste und die Applikations-Frontends verknüpft (Konnektivität, Krafzig/Banke/Slama 2006, 65), um darüber Dienste aufrufen zu können. Ein solcher Dienstbus basiert im Gegensatz z.B. zu CORBA (siehe dazu Abschnitt 2.3.4.2) nicht notwendigerweise nur auf einer Technologie, sondern kann mehrere Produkte und Konzepte, z.B. CORBA oder RMI für asynchrone Kommunikation (Krafzig/Banke/Slama 2006, 162), umfassen. Darunter zählt auch die Verknüpfung unterschiedlicher Betriebssysteme, Programmiersprachen, Laufzeitumgebungen, Middlewares oder Kommunikationsprotokolle (Integration heterogener Technologien, Krafzig/Banke/Slama 2006, 65). Dieser Dienstbus sollte somit zumindest synchrone und asynchrone Kommunikation (Integration heterogener Kommunikationskonzepte, Krafzig/Banke/Slama 2006, 65) unterstützen. Neben der Verknüpfung der Komponenten in einer SOA muss der Dienstbus auch Funktionalitäten für z.B. Transaktionen, Nachrichtentransformation oder Sicherheit (technische Dienste, Krafzig/Banke/Slama 2006, 65) anbieten.

Im Dienstverzeichnis werden die Contracts der Dienste der SOA gespeichert, um ihr Auffinden zu ermöglichen. Die auf diese Weise bekannt gemachten Dienste von Komponenten können von anderen Elementen gesucht und verwendet werden. Dazu müssen die Schnittstellen der Dienste standardisiert spezifiziert werden.

Die wesentlichen Eigenschaften einer SOA können, zusammengefasst aus den bisherigen Ausführungen, wie folgt beschrieben werden (Richter/Haller/Schrey 2005, 413):

- Architekturmuster zum Aufbau einer Informationssystemlandschaft aus einzelnen fachlichen Anwendungsbausteinen mit klar spezifizierten Funktionalitäten
- Lose Kopplung der Applikationsbausteine durch die Bereitstellung der Aufgaben über Dienste
- Ein Dienst kann als eine Komponente verstanden werden, die den spezifizierten Dienst dem Dienstnutzer zur Verfügung stellt. Die Implementierung der Dienste ist hinter einer wohl definierten Schnittstelle verborgen. Ein Dienst kann mit anderen Diensten zu Verarbeitungsabläufen kombiniert werden.
- Der Aufruf der Dienste erfolgt plattformunabhängig über einen einheitlichen Mechanismus, der von der existierenden Kommunikationsinfrastruktur abstrahiert.

Die in Abbildung 3.2-5 dargestellten Architekturelemente lassen sich entsprechend der Klassifikation bei *Schelp/Schwinn* (2005), *Erl* (2005), *Alonso et al.* (2004) und *Legner/Heutschi* (2007, 1644-1645) den für eine SOA typischen Schichten zuordnen bzw. werden von diesen verfeinert: Application System, Services, Workflow Integration und Desktop Integration. Die Zuordnung zwischen SOA-Architekturelementen und -schichten ist in Abbildung 3.2-6 dargestellt und wird im Folgenden beschrieben (siehe dazu auch die Ausführungen bei Legner/Heutschi 2007):

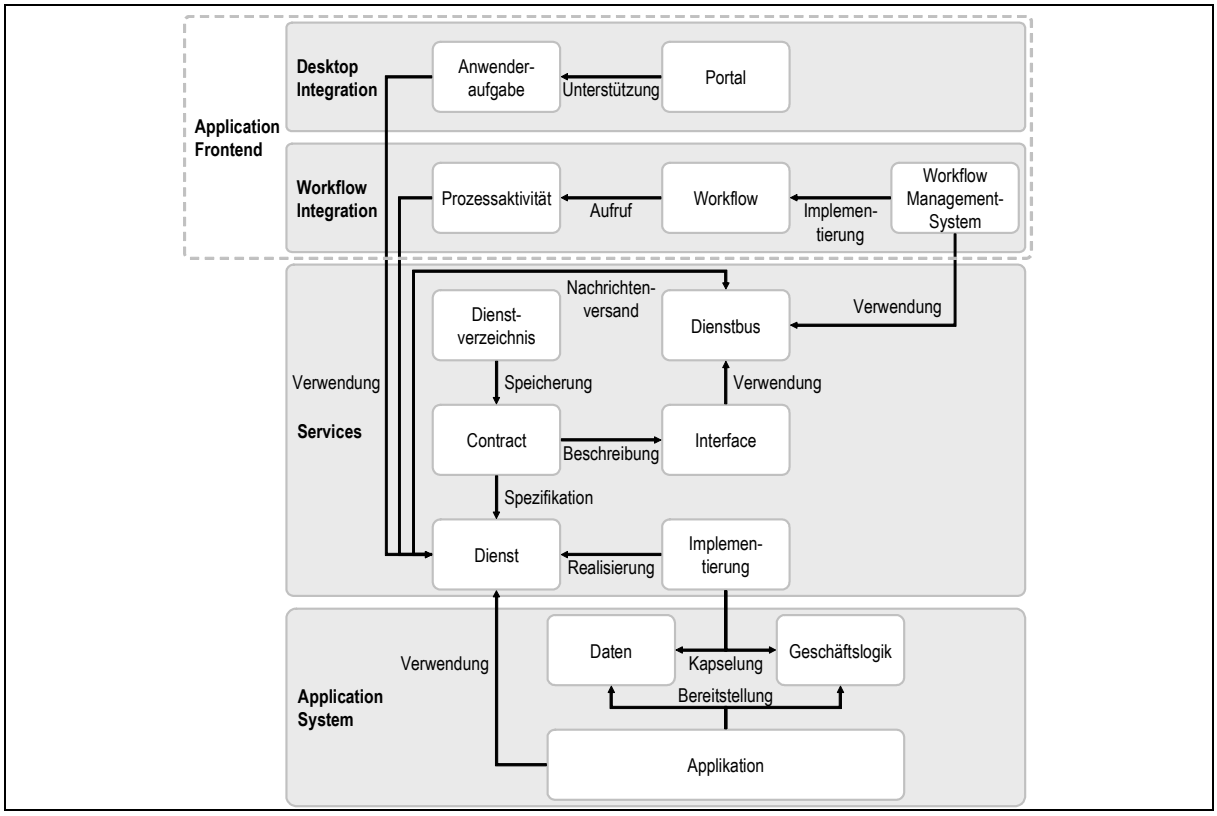

Abbildung 3.2-6: ***Architekturschichten einer SOA***
Quelle: Eigene Darstellung, in Anlehnung an *Legner/Heutschi* (2007, 1645)

Die zu integrierenden Applikationen werden der Ebene Application System zugeordnet und sind damit im Gegensatz zu den Ausführungen bei *Krafzig/Banke/Slama* (2006, 55-66) in der Klassifikation in Abbildung 3.2-6 explizit genannt. Funktionalitäten mit Bezug zu Diensten werden in der Schicht Services zusammengefasst. Die bei *Krafzig/Banke/Slama* (2006, 55-66) fehlende Detaillierung des Application Frontends wird durch die Ebenen Workflow Integration und Desktop Integration verfeinert. Über die Workflow Integration werden die Geschäftsprozesse gesteuert. Dazu werden zur Erfüllung der einzelnen Aktivitäten Dienste aufgerufen. Zur Bearbeitung von Anwenderaufgaben werden ebenso geeignete Dienste aktiviert. Die Benutzerschnittstelle ist oftmals so gestaltet, dass jeweils dedizierte Sichten auf die Informationen zur Verfügung gestellt werden. Realisiert werden kann dies z.B. durch ein Portalsystem (siehe dazu Abschnitt 6.7.5.1).

Beispiele zur technischen Umsetzung einer SOA sind Web Services (Kossmann/Leymann 2004), CORBA (Richter/Haller/Schrey 2005, 415; Hasselbring 2006, 49) oder auch Werkzeu-

ge zur EAI (Richter/Haller/Schrey 2005, 415). Die Realisierung über CORBA oder EAI-Werkzeuge erfordert jedoch meist die individuelle Konstruktion einer SOA-Infrastruktur sowie die Adaption der Applikationsbausteine an die Integrationsplattform (Richter/Haller/Schrey 2005, 415). Mit der zunehmenden Verbreitung und Standardisierung von Web Services (siehe dazu Abschnitt 3.2.4.4) wird jedoch der Wegfall dieser zusätzlichen Aufgaben erwartet (Richter/Haller/Schrey 2005, 415). Deshalb kann Web Services zur Realisierung einer SOA ein wesentliches Potenzial attribuiert werden.

Insgesamt abstrahiert eine SOA von der technischen Implementierung (Krafzig/Banke/Slama 2006, 57) und fokussiert die Unterstützung von Geschäftsprozessen. In einer SOA werden somit die technischen Details verborgen und die fachlichen Zuständigkeiten getrennt, womit etablierte Konzepte der Software-Architektur auf Anwendungslandschaften übertragen werden (Richter/Haller/Schrey 2005, 413). Als wesentliche Konzepte einer SOA gelten die Prozessorientierung sowie die lose Kopplung (Humm/Voß/Hess 2006, 410). Dienste werden entsprechend der zu realisierenden Geschäftsfunktionalität vertikal und leichtgewichtig implementiert. Weiterhin kann festgestellt werden, dass sowohl EAI als auch SOA bei der Integration von Informationssystemen eine bedeutende Rolle einnehmen. Die Differenzierung dieser beiden Ansätze wird im folgenden Abschnitt 3.2.4.3 vorgenommen.

3.2.4.3 Abgrenzung zwischen EAI und SOA

EAI kann als ein Treiber für die Einführung einer SOA betrachtet werden (Krafzig/Banke/Slama 2006, 215). Somit kann eine EAI als ein möglicher Bestandteil zur Realisierung einer SOA aufgefasst werden. Eine trennscharfe Differenzierung zwischen EAI und SOA kann deshalb oftmals nicht vorgenommen werden. Daher können in der Praxis meist Mischformen aus beiden Ansätzen vorgefunden werden.

Für eine mögliche Abgrenzung zwischen einer EAI und einer SOA werden im Folgenden wesentliche Differenzierungsmerkmale identifiziert und jeweils beschrieben (siehe Tabelle 3.2-1):

Der Fokus der EAI liegt hinsichtlich des Kriteriums Integrationsradius eher auf einem einzigen Unternehmen, wobei auch einzelne Unternehmenspartner oder Niederlassungen über Adaptionspunkte angeschlossen werden können. Im Gegensatz dazu zielt eine SOA eher auf die Integration über Unternehmensgrenzen hinweg ab.

Hinsichtlich Integrationstechnologien wird bei der EAI die Umsetzung über ein einziges EAI-Produkt angestrebt. Mit einem solchen Produkt werden die zu integrierenden Systeme gekoppelt. Ggf. können auch mehrere EAI-Produkte – aggregiert zu einer EAI-Plattform – eingesetzt werden, wobei diese meist spezialisierte Aufgaben wahrnehmen und so z.B. synchrone oder asynchrone Kommunikation unterstützen. Bei einer SOA hingegen erfolgt eine Abstraktion von den einzelnen Produkten und somit die Fokussierung auf Service-Schnittstellen. Bedeutend ist hier der Einsatz von Standards auf der jeweiligen Protokollebene. Zudem lässt sich konstatieren, dass eine homogene Integration bei einer SOA oft gar nicht möglich ist, weil bestimmte Informationssysteme eine definierte EAI-Plattform voraussetzen. So ist z.B. ein SAP-System an die EAI-Plattform SAP XI (Exchange Infrastructure, für weitere Ausführungen dazu siehe Stumpe/Orb 2005; Nicolescu et al. 2006) gebunden.

Kriterium	EAI	SOA
Integrationsradius	Fokus auf Integration innerhalb eines einzigen Unternehmens	Fokus auf Integration über Unternehmensgrenzen hinweg
Integrationstechnologien	Fokus auf ein einziges EAI-Produkt	Transparenter Einsatz der Menge der gewählten Integrationskomponenten
Integrationsfunktionalität	Bereitstellung der Funktionalität auf technischer Ebene	Bereitstellung der Funktionalität durch Dienste mit wohl definierter Schnittstelle
Integrationselemente	Daten, Objekte, Prozesse	Daten, Geschäftsobjekte, Prozesse
Architekturparadigma	Zentraler Message Broker in monolithischer Schichtenarchitektur	Leichtgewichtige, vertikal implementierte Dienste, Kommunikation über einen Dienstbus
Komplexität	Funktionalität zur Integration vollständig im Message Broker	Funktionalitäten können aus den jeweiligen Diensten komponiert werden, womit die Reduktion der Komplexität (Krafzig/Banke/Slama 2006, 70) gegeben ist
Datenmodell	Monolithisch (Krafzig/Banke/Slama 2006, 72)	Vertikal, in Abhängigkeit von der Geschäftsentität
Präsentationsschicht	Nicht gegeben (Strüver 2006, 18)	Desktop Integration
Ziel	Integration einer heterogenen Informationssystemlandschaft	Integration einer heterogenen Informationssystemlandschaft, Wiederverwendung (Krafzig/Banke/Slama 2006, 70), Flexibilität

Tabelle 3.2-1: ***Vergleich von EAI und SOA***
Quelle: Eigene Darstellung

EAI fokussiert zur Bereitstellung von Integrationsfunktionalität den Aufruf der Funktionalitäten auf technischer Ebene z.B. mit der eindeutigen Spezifikation der Datentypen und Methodensignatur. Fehlermeldungen müssen dementsprechend auch auf dieser an der Implementierung orientierten Ebene behandelt werden. Dienste bei einer SOA stellen sich, vergleichbar mit abstrakten Datentypen (ADT), nach außen als geschlossen dar. Das interne Format der Datenstruktur bzw. der Realisierungen ist verborgen, weil bei SOA eine wohl definierte Schnittstelle auf abstrakter Ebene angegeben wird. Die Fehlerbehandlung bei einem Dienstaufruf in einer SOA ist folglich an eine solche Fehlerbehandlung angelehnt, wie sie in der Objektorientierung mit Ausnahmen (siehe Abschnitt 2.2.5) eingeführt wurde.

Hinsichtlich des Kriteriums Integrationselemente können bei EAI Daten, Objekte und Prozesse identifiziert werden. Bei der SOA liegt der Fokus auf Diensten, die gekapselt und über eine wohl definierte Schnittstelle die jeweils erforderlichen Aktionen transparent ausführen. Dazu stehen für Daten datenzentrierte, für Objekte logikzentrierte und für Prozesse prozesszentrierte Dienste zur Verfügung. Obwohl die semantische Ähnlichkeit zwischen den Integrationselementen bei einer EAI bzw. SOA offensichtlich ist, erfolgt der Aufruf der Dienste über

einheitliche Schnittstellen. Letztere sind in einer EAI nicht gegeben, weil je nach Technologie Funktionalitäten mit unterschiedlicher Aufrufschnittstelle zur Verfügung gestellt werden.

Obwohl jeweils bei einer EAI und SOA die Funktionalität eines Message Brokers bzw. Dienstbus identifiziert werden kann, folgt ihre Implementierung dennoch unterschiedlichen Architekturparadigmen: In einer EAI realisiert der zentrale Message Broker die Aufgaben der Integration in einer monolithischen Weise. Die bei einer EAI eingesetzten Integrationstechnologien nur in horizontaler Dimension strukturiert. Bei einer SOA hingegen werden die Dienste in vertikaler Richtung implementiert und stellen damit auf der jeweiligen Schicht nur einen bestimmten Funktionalitätsausschnitt zur Verfügung, der sich an Geschäftsobjekten oder -vorgängen orientiert.

Die Komplexität der Implementierung ist bei einer EAI umfangreich, weil die Aufgaben des Message Brokers in vollständig horizontaler Dimension implementiert sind. Im Gegensatz dazu wird die Funktionalität in einer SOA über mehrere leichtgewichtige Dienste komponiert, die jeweils, bezogen auf eine bestimmte Geschäftsentität, vertikal implementiert sind. Damit kann die Reduktion der Komplexität (Krafzig/Banke/Slama 2006, 70) für die Implementierung der Funktionalitäten des Dienstbus erzielt werden.

Das Datenmodell in einer EAI ist typischerweise monolithisch gestaltet (Krafzig/Banke/Slama 2006, 72). Dazu wird z.B. über eine Zugriffsschicht für eine Datenbank auf das vollständige Datenmodell mit seinen expliziten und impliziten Abhängigkeiten zugegriffen (Krafzig/Banke/Slama 2006, 72). Im Gegensatz dazu ist das Datenmodell in einer SOA vertikal angelegt (Krafzig/Banke/Slama 2006, 72): Dazu werden den jeweiligen Diensten in Abhängigkeit von den Geschäftsentitäten die Verantwortlichkeiten für die zugehörigen Daten übertragen.

Durch eine EAI ist eine Präsentationsschicht nicht gegeben (Strüver 2006, 18). Ihr Fokus liegt lediglich auf der Integration der Informationssysteme mit ihren bereits vorgegebenen Benutzerschnittstellen. Bei einer SOA ist in der Desktop Integration hingegen eine dedizierte Präsentationsschicht bestimmt, die zusätzlich die jeweiligen Bedürfnisse des Anwenders über die Bereitstellung unterschiedlicher Sichten auf die Daten der Informationssysteme berücksichtigt.

Eine EAI besitzt als primäres Ziel die Integration einer heterogenen und gewachsenen Anwendungslandschaft. Obwohl auch eine SOA auf die Integration von gegebenen Informationssystemen ausgelegt ist, ist dabei ein wesentliches Ziel die Wiederverwendung und die Reduktion von Implementierungskomplexität (Krafzig/Banke/Slama 2006, 70). Dies gelingt in einer SOA mit ihren Diensten in besonderer Weise, denn diese sind so gestaltet, dass sie zu kompositen Diensten zusammengestellt werden können und damit in unterschiedlichen Kontexten Anwendung finden.

Zusammenfassend kann festgestellt werden, dass eine EAI die Integration von heterogenen, gewachsenen Informationssystemlandschaften fokussiert, die über einen Message Broker realisiert wird. In dem SOA-Konzept wird darüber hinaus mit der Dienstabstraktion entlang relevanter Geschäftsentitäten die Reduktion der Komplexität des Dienstbus sowie die Wiederverwendung auf der Dienstebene angestrebt. Insgesamt kann die Integration

durch eine EAI somit als datenbezogen betrachtet werden, während eine SOA Dienste fokussiert. Damit kann einer SOA eine über eine EAI hinausgehende Flexibilität attribuiert werden.

3.2.4.4 Web Services

Web Services können als eine geeignete Basis für eine Umsetzung der EAI (Kossmann/Leymann 2004, 117) und damit auch, entsprechend den Ausführungen in Abschnitt 3.2.4.3, für die Realisierung einer SOA betrachtet werden. Deshalb werden im Folgenden, ausgehend von der Beschreibung der Eigenschaften von Web Services, die für eine Umsetzung erforderlichen Technologien dargestellt. Web Services werden bei *Kossmann/Leymann* (2004) wie folgt definiert:

> [Web Services sind ein] „Bündel von Technologien zur Beschreibung von Schnittstellen und Eigenschaften von Implementierungen der Schnittstellen, Beschreibung von Datenaustauschformaten und Qualitätseigenschaften des Austauschs, Registrierung von Komponenten, Komposition von Komponenten und Sicherheit im Austausch von Komponenten“ (Kossmann/Leymann 2004, 117).

Als Ergänzung dazu lassen sich folgende Eigenschaften von Web Services identifizieren (Kossmann/Leymann 2004, 119):

- *Lose Kopplung*: Nur unmittelbare Interaktion von Anwendungen über einen Message Broker
- *Virtualisierung*: Realisierung über ein Regelwerk im Message Broker
- *Einheitliche Konventionen* und *Standards*: Die Kommunikation über den Message Broker setzt die Verwendung von Konventionen und Standards voraus.

Aus technischer Perspektive kann die Definition für einen Web Service weiter konkretisiert werden (Kossmann/Leymann 2004, 120):

- Identifikation eines Web Services über einen Uniform Resource Identifier (URI)
- Die mit WSDL (siehe weiter unten) beschriebene Schnittstelle eines Web Services ist maschinenlesbar.
- Die Kommunikation eines Web Services erfolgt über XML-Nachrichten (Extensible Markup Language), die auf Internetprotokollen basieren.
- Die Autonomie von Web Services wird durch die eigenständige Verarbeitung von empfangenen Nachrichten erreicht. Qualitätseigenschaften der Dienste sind separat zu spezifizieren.

Für die Implementierung einer Informationssystemintegration auf der Basis von Web Services wird ein definierter Protokoll-Stack (siehe Abbildung 3.2-7) eingesetzt, dessen Bestandteile Netzwerkprotokolle, XML, SOAP, WSDL und UDDI im Folgenden beschrieben werden: Mit SOAP (Simple Object Access Protocol, Mitra 2003) erfolgt die Spezifikation der Grobstruktur und der Verarbeitungsvorschriften von Nachrichten (Kossmann/Leymann 2004, 120). Diese Beschreibung basiert auf XML und ist damit unabhängig von Implementierungs- und Plattformentscheidungen (Kossmann/Leymann 2004, 120). Mit SOAP kann die Spezifikation der genannten Eigenschaften somit flexibel erfolgen (Kossmann/Leymann 2004, 120-121):

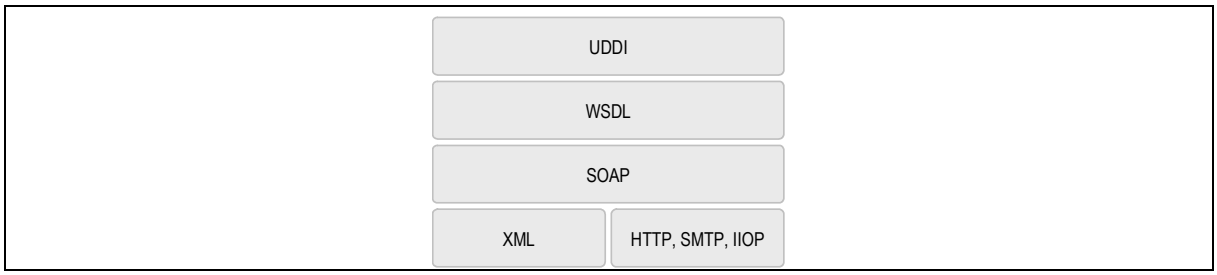

Abbildung 3.2-7: ***Protokoll-Stack für Web Services***
Quelle: Eigene Darstellung, in Anlehnung an *Kossmann/Leymann* (2004, 120)

- Neben XML-Daten können auch binäre Dateien in eine SOAP-Nachricht verpackt werden.
- SOAP kann mit verschiedenen Netzwerkprotokollen wie HTTP (auf der Basis von HTTP POST oder HTTP GET, Krafzig/Banke/Slama 2006, 22), SMTP (Simple Mail Transfer Protocol) oder IIOP verwendet werden.
- Unterstützung unterschiedlicher Interaktionsprotokolle zwischen Diensterbringer und Dienstnutzer

Mit WSDL (Web Services Description Language, Christensen et al. 2001) wird die Schnittstelle für einen Web Service beschrieben (Kossmann/Leymann 2004, 121). Damit werden die folgenden Bestandteile von Web Services in XML spezifiziert (Kossmann/Leymann 2004, 121):

- Deklaration von Operationen
- Definition der zu sendenden und zu empfangenden Nachrichten
- Spezifikation der Protokolle für den Nachrichtenaustausch und der Kodierung
- Benennung und Angabe der Adresse eines Web Services

Um die Dienstqualität über WSDL hinaus zu beschreiben, können Policies eingesetzt werden (Kossmann/Leymann 2004, 122), deren Eigenschaften im Folgenden beschrieben werden (Kossmann/Leymann 2004, 122-123):

Policies werden in einer eigenen Sprache spezifiziert. Dadurch können Eigenschaften wie Transaktionsunterstützung, Sicherheit, Verschlüsselung, Signatur oder weitere Informationen wie die Kosten oder die Zahlungsart für die Benutzung eines Dienstes angegeben werden. Diese Policies können mit einzelnen Elementen der WSDL-Definition assoziiert werden und dabei auch flexibel nach dem Deployment der Web Services zugewiesen werden.

Im UDDI (Universal Description, Discovery and Integration) wird ein zentrales Dienstverzeichnis realisiert. Letzteres leistet damit einen Beitrag zum Management von Web Services und zur Virtualisierung der angebotenen Dienste sowie Ressourcen (Kossmann/Leymann 2004, 122). Die Dienste im UDDI werden in drei unterschiedlichen Beschreibungsformen vorgehalten (Kossmann/Leymann 2004, 122): Green Pages, White Pages und Yellow Pages. In den Green Pages sind die WSDL-Schnittstellen der registrierten Dienste hinterlegt (Kossmann/Leymann 2004, 122). In den White Pages kann eine Suche mit einer genauen Dienst-

spezifikation erfolgen, in Yellow Pages entlang einer Funktionalitätsbeschreibung des gewünschten Dienstes.

Abbildung 3.2-8 beschreibt den Zusammenhang der Komponenten Client, Provider und UDDI sowie ihre Interaktion zur Registrierung, Identifikation und Nutzung eines Dienstes: Der Diensterbringer registriert seinen Dienst im UDDI. Ein Client sucht in dem Dienstverzeichnis nach einem geeigneten Dienst. Die Interaktion zwischen Client und Diensterbringer erfolgt nach der Identifikation des passenden Dienstes direkt.

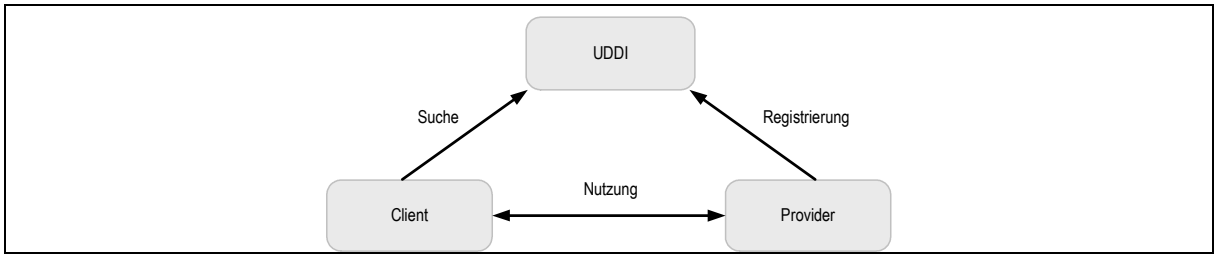

Abbildung 3.2-8: ***Registrierung, Suche und Nutzung eines Web Services***
Quelle: *Kossmann/Leymann* (2004, 122)

Zusammenfassend lässt sich feststellen, dass Web Services eine geeignete Technologie darstellen, die wegen ihrer Plattformunabhängigkeit, der Verwendung von Standards sowie ihrer losen Kopplung für die Integration von Informationssystemen im Sinne der EAI bzw. SOA eingesetzt werden kann. Im Sinne einer SOA wird ein Web Service über den Protokoll-Stack XML, z.B. HTTP und SOAP implementiert. Der Contract eines Dienstes wird durch Policies spezifiziert. Die Schnittstellenbeschreibung ist über WSDL gegeben. Damit erfüllt ein Web Service die Eigenschaften eines Dienstes.

Die bisher im Abschnitt 3.2 beschriebenen Architekturen werden im folgenden Abschnitt 3.3 jeweils aus der Perspektive einer agentenbasierten Realisierung betrachtet.

3.3 Agentenbasierte Architekturen

Die Analyse, das Design und die Implementierung von komplexen Software-Systemen als eine Gesellschaft von interagierenden, autonomen Agenten erlauben dem Software-Entwickler signifikante Vorteile gegenüber herkömmlichen Methoden (Jennings 2001, 35). Dazu zeigt *Jennings* (2001, 36-41) auf, wie sich die besonderen Eigenschaften von Agenten einsetzen lassen, um solche Systeme vorteilhaft zu realisieren. Diese Argumentationslinie wird im Folgenden über die Darstellung der kanonischen Sicht eines komplexen Systems, der besonderen Eigenschaften von Agenten und der Konstruktion eines komplexen Systems mit Agenten beschrieben. Diese Ausführungen dienen als Grundlage für einen Vergleich von agentenbasierten Architekturen mit den in den Abschnitten 3.2.3 und 3.2.4 dargestellten Architekturen.

Die kanonische Sicht eines komplexen Systems (Booch 1994, 13-15) besitzt nach *Booch* (1994, 16-21) die folgenden wesentlichen Eigenschaften:

- *Dekomposition*: Um komplexe Systeme erfassbar zu gestalten, werden diese in mehrere Subsysteme strukturiert. Damit können für die Entwicklung eines Systems die relevanten Teile fokussiert werden.

- *Abstraktion*: Um die Komplexität zu reduzieren, erfolgt die Konzentration auf die für den jeweiligen Zweck relevanten Eigenschaften.
- *Hierarchie*: Zwischen den Elementen eines komplexen Systems werden Interaktionsmechanismen beschrieben. Die Komplexität wird hier reduziert, indem aus atomaren Komponenten Subsysteme gebildet werden und damit die Interaktionen auf der Ebene der Subsysteme betrachtet werden können (Jennings 2001, 36).

Zur Ableitung eines agentenbasierten, komplexen Systems aus dieser Perspektive werden zunächst diejenigen Eigenschaften beschrieben, durch die sich agentenbasierte Systeme von herkömmlichen Systemen unterscheiden (Jennings 2001, 37):

- Die Kommunikation zwischen Agenten erfolgt auf abstrakter Ebene in Form von semantisch reichhaltigen Nachrichten.
- Diese Interaktionen sind flexibel von den Agenten zu bearbeiten, d.h. sie müssen in der Lage sein, abhängig vom jeweiligen Kontext auf die empfangenen Nachrichten zu reagieren.
- In agentenbasierten Systemen können Agentengruppen flexibel gebildet werden (Jennings 2001, 37), die zur Laufzeit des Systems verändert werden können (Odell/Nodine/Levy 2004). Somit sind Agentensysteme durch komplexe und sich verändernde Organisationsbeziehungen und damit auch Architekturen zwischen den Agenten gekennzeichnet.

Mit diesen besonderen Eigenschaften von Agenten kann die kanonische Sicht eines Multiagentensystems nach *Jennings* (2001, 37) wie folgt beschrieben werden: Die Problemstellung wird in viele, autonome Komponenten zerlegt, die flexibel agieren und interagieren können, um ihre Ziele zu erreichen. Die Abstraktionen in agentenbasierten Systemen sind dabei Agenten, Gruppen von Agenten, Interaktionen und sich dynamisch ändernde Organisationen. Weiterhin erweist sich die dynamische Bildung von Organisationsbeziehungen in einem Agentensystem als geeignet, um die Abhängigkeiten und Interaktionen adäquat abzubilden, die in einem komplexen System zu beobachten sind (Jennings 2001, 37).

Zusammenfassend lässt sich mit Bezug auf die allgemeinen Eigenschaften eines komplexen Systems zeigen, dass Agentensysteme die Eigenschaften dieser Systeme vorteilhaft abbilden können und damit in der Lage sind, die Komplexität von Software-Systemen zu reduzieren (Jennings 2001, 37-39):

- *Dekomposition*: Agenten werden entsprechend ihren Zielen gruppiert. Diese Kopplung erfolgt auf semantischer Ebene, weil die Interaktionen nachrichtenbasiert sind. Weiterhin besitzen Agenten jeweils einen eigenen Kontrollfluss.
- *Abstraktion*: Die wesentlichen Elemente in komplexen Systemen, Subsysteme, Komponenten, Interaktionen und Organisationsbeziehungen, können unmittelbar auf das Agentenparadigma übertragen werden.
- *Hierarchie*: Die hierarchische Relation zwischen den Komponenten eines komplexen Systems in der Form von Agenten kann aus unterschiedlichen Perspektiven betrachtet werden: Ein Subsystem aus Agenten verhält sich nach außen wie eine einzige Komponente (siehe dazu auch die Ausführungen in Abschnitt 2.3.3).

Insgesamt wird deutlich, dass sich komplexe Systeme auf der Basis von Software-Agenten vorteilhaft konstruieren lassen.

Bei der Betrachtung eines Multiagentensystems aus der Perspektive einer Organisationsstruktur kann die Relation zu etablierten Architekturmustern, wie sie in den Abschnitten 3.2.3 und 3.2.4 beschrieben sind, hergestellt werden. Damit eignen sich Software-Agenten als elementare Einheit der Software-Technik, um in unterschiedliche Architekturen hineinentwickelt zu werden. Dies ist auch das Prinzip, dem Reinke (2003) bei der architekturbasierten Konstruktion von Agentensystemen (siehe dazu auch Abschnitt 1.7) folgt. Diese Beziehungen zwischen Agenten und Architekturen werden in den folgenden Abschnitten 3.3.1 mit 3.3.8 dargestellt.

3.3.1 Agenten und Schichtenarchitekturen

Eine Schichtenarchitektur auf der Basis von Agenten kann konstruiert werden, indem die jeweiligen Agenten entsprechend ihren Aufgaben einer geeigneten Schicht zugeordnet werden. Insbesondere kann eine agentenbasierte Anwendung über die Schichten Persistenz, Applikationslogik und Visualisierung sowie Benutzerinteraktion konstruiert werden. Die Strukturierung auf der Basis von Schichten bestimmt somit eine Metaarchitektur, in die das Agentensystem hineinentwickelt wird. Dieses Prinzip wird auch in der vorliegenden Arbeit realisiert.

3.3.2 Agenten und verteilte Architekturen

Agentenbasierte Systeme sind per se als verteilte Systeme zu betrachten. Agentenplattformen bieten die für die Interaktion der agentenbasierten Applikationen erforderlichen Infrastrukturdienste an. Somit können agentenbasierte Systeme als eine Implementierungsmöglichkeit eines verteilten Systems betrachtet werden.

Client-Server-Architekturen zeichnen sich im Wesentlichen durch eine Aufteilung in Clients und Server aus. Diese Rollenvergabe ist meist fest vorgegeben und führt zu einer starren Architektur. Dabei sind die Schnittstellen zwischen Client und Server meist klar definiert, womit eine enge Kopplung (siehe Abschnitt 3.4.1) zwischen den Komponenten gegeben ist. Mit einer agentenbasierten Realisierung einer Client-Server-Architektur ergeben sich die folgenden Differenzierungsaspekte: Je nach Implementierung können Software-Agenten die Rolle von Clients oder Servern sowie Doppelrollen innehaben. Weiterhin können zum gleichen Zeitpunkt von einem Agenten mehrere Rollen eingenommen werden. Wenn Dienste von mehreren Agenten angeboten werden, kann zur Laufzeit eine dynamische Skalierung des Systems erfolgen, indem bei einer möglichen Auslastung weitere Agenten hinzugezogen werden, die in die Gruppe der Agenten eingebunden werden, die die Anfragen bearbeiten. Damit ist die Architektur einer flexiblen Adaption unterworfen. Durch den Austausch semantisch reichhaltiger Nachrichten ist eine vollständig lose Kopplung (siehe Abschnitt 3.4.1) in dem verteilten System gegeben.

Damit wird deutlich, dass ein verteiltes System durch Software-Agenten vorteilhaft realisiert werden kann. Um die in einer Client-Server-Architektur inhärenten Problemstellungen (siehe dazu Abschnitt 3.2.3.4) zu beseitigen, kann ein agentenbasiertes System zu einem Peer-to-Peer-Netzwerk ausgebaut werden. Dies wird im folgenden Abschnitt 3.3.3 beschrieben.

3.3.3 Agenten und ambiente Intelligenz

Obwohl Software-Agenten aus den in Abschnitt 2.3.9 genannten Gründen noch nicht weit verbreitet sind, können bereits Zusammenhänge mit noch in der Forschung befindlichen, an-

deren Gebieten hergestellt werden. Im Folgenden wird, ausgehend von einer Begriffsbildung zur ambienten Intelligenz, die Möglichkeit der Realisierung der ambienten Intelligenz mit Agenten beschrieben:

Werden agentenbasierte Systeme mit zusätzlichen Discovery Middlewares (siehe dazu weiter unten) erweitert, können damit Peer-to-Peer-Architekturen realisiert werden, die auch als Basis für Ad-hoc-Netzwerke in der ambienten Intelligenz (siehe Abschnitt 3.3) betrachtet werden können. Dabei ist die Fähigkeit von Agenten vorteilhaft, auf redundante Weise Dienste anzubieten, die über ein Dienstverzeichnis von anderen Agenten aufgerufen werden können.

Der Begriff ambiente Intelligenz wird bei *Pirker/Berger/Watzke* verstanden als die Kombination der folgenden Eigenschaften (2004):

Ambiente Intelligenz zeichnet sich aus durch intelligente persönliche Assistenz und Delegation sowie die Fähigkeit, Dienste in einer digitalen Ambienz dynamisch zu finden und mit diesen zu interagieren.

Mit dieser Definition wird auch der Übergang zum proaktiven Computing (siehe auch Tennenhouse 2000) deutlich, weil von solchen Systemen der Bedarf des Benutzers antizipiert und dessen Bereitstellung proaktiv angegangen wird (Pirker/Berger/Watzke 2004). Voraussetzung für die Realisierung der ambienten Intelligenz ist die Existenz des ubiquitären Computings (Pirker/Berger/Watzke 2004). In Anlehnung an *Weiser* (1991) definiert *Krcmar* (2005, 506) diesen Begriff wie folgt (für weitere Ausführungen zum Ubiquitous Computing siehe Krcmar 2005, 504-515):

> **„Ubiquitous Computing [im Original fett] zielt ab auf eine verbesserte Computernutzung durch die allgegenwärtige Bereitstellung von Rechnern in der physischen Umgebung. Die Computer verschwinden weitestgehend aus dem Sichtfeld der Anwender".**

Agentensysteme eignen sich prinzipiell für die Umsetzung der ambienten Intelligenz (Pirker/Berger/Watzke 2004; Bauer/Odell 2005). Dazu sind jedoch Erweiterungen von etablierten Agentenplattformen notwendig: Die für die ambiente Intelligenz erforderliche Dienstidentifikation in einer regulären FIPA-Agentenplattform ist zu wenig flexibel, weil zunächst ein entsprechender DF und anschließend darin ein passender Dienst gefunden werden muss (Pirker/Berger/Watzke 2004). Um etablierte Agentenplattformen mit den erforderlichen Eigenschaften zur Dienstfindung und Interaktion für Ad-hoc-Netzwerke zu erweitern, wie sie in der ambienten Intelligenz erforderlich sind, wird auf der Basis von bereits vorhandenen Discovery Middlewares bei *Pirker/Berger/Watzke* (2004) ein entsprechender Ansatz vorgeschlagen. Dazu wird die FIPA-Agentenplattform (siehe Abschnitt 2.3.4.1) mit einem Dienst erweitert, der auf JXTA (Akronym für juxtapose, CollabNet Inc. 2006) basiert und somit Ad-hoc-Netzwerke unterstützt. JXTA definiert eine Familie von Protokollen zur Realisierung von Ad-hoc-Netzwerken (CollabNet Inc. 2006). Diese Technologie bringt die Vorteile der Skalierbarkeit sowie die Kombination von Multicasting mit verteilten Hash-Tabellen (siehe dazu insbesondere Abschnitt 3.2.3.4) mit sich (Pirker/Berger/Watzke 2004). Die beschriebene Erweiterung der FIPA-Plattform ist so gestaltet, dass auch andere Discovery Middlewares

eingesetzt werden können (Pirker/Berger/Watzke 2004). Somit ist die Agentenplattform nicht auf einzelne Discovery Middlewares eingeschränkt, sondern kann nach Bedarf angepasst werden.

Über die beschriebene Extension einer Agentenplattform mit Peer-to-Peer-Funktionalität werden agentenbasierte Systeme zu einem System mit ambienter Intelligenz erweitert. Letztendlich findet hier eine Synthese der Vorteile von Peer-to-Peer-Netzwerken zur Realisierung von verteilten Systemen und von Agentensystemen statt.

Der Aspekt der Selbstorganisation, wie er bereits in der ambienten Intelligenz mit der Antizipierung des Informationsbedarfs gegeben ist, wird mit dem Konzept des Organic Computing verstärkt, indem dort eine weitere Abstraktionsschicht eingeführt wird. Im folgenden Abschnitt wird insbesondere der Zusammenhang zwischen dem Organic Computing und Agenten dargestellt.

3.3.4 Agenten und Organic Computing

Eine Anwendung, die den Prinzipien des Organic Computing folgt, kann geeignet durch ein agentenbasiertes System realisiert werden. Ausgehend von der Darstellung der Eigenschaften von selbstorganisierenden Systemen wird im Folgenden beschrieben, wie ein technisches System mit den Charakteristika des Organic Computings realisiert werden kann. Es folgt eine Darstellung einer Umsetzungsmöglichkeit auf der Basis von Agenten:

Das Wesen des Organic Computing (auch bekannt unter der Bezeichnung Autonomic Computing, Kephart/Chess 2003; Sterritt 2005) ist die Übertragung von Eigenschaften natürlicher auf technische Systeme, um deren Komplexität beherrschen zu können (Rochner/Müller-Schloer 2005, 195). Natürliche Systeme sind „selbstorganisierend, -konfigurierend, -optimierend, -heilend und -schützend“ (Müller-Schloer 2005, 179; zu den Eigenschaften von selbstorganisierenden Systemen siehe auch Kephart/Chess 2003, 42-43; Sterritt 2005, 80 und 85). Solche Systeme sind in der Lage, sich flexibel an sich ändernde Umweltbedingungen anzupassen (Müller-Schloer 2005, 179). Die dafür eingesetzte Selbsterhaltung und eigenständige Reproduktion von natürlichen Organismen wird von *Maturana* (1982) als *Autopoiese* bezeichnet. Sich selbst organisierende, natürliche Systeme sind damit durch das Zusammenwirken von teilautonomen Elementen gekennzeichnet, wobei meist auf eine zentrale zugunsten einer dezentralen Steuerung verzichtet wird, Elemente zu Subsystemen zusammenwachsen und das Prinzip der „Lokalität der Kommunikation“ (Müller-Schloer 2005, 179) gilt (Müller-Schloer 2005, 179). Demnach agieren Elemente eigenständig, wobei sie zusätzlich in ein Ganzes integriert werden (Müller-Schloer 2005, 179). Einheiten mit einer solchen Doppelrolle werden von *Koestler* (1984) und *Popper* (1989) als *Holonen* bezeichnet. Eine zentrale Bedeutung in selbstorganisierenden Systemen nimmt die *Emergenz* ein (Rochner/Müller-Schloer 2005, 196). Darunter wird die Fähigkeit eines Systems verstanden, aus ursprünglich chaotischen Verhältnissen ein geordnetes System zu entwickeln, wobei die Gesamtheit des Systems mehr als die einfache Summe der Individuen erreicht (Rochner/Müller-Schloer 2005, 196).

Bislang werden Systeme meist nach dem Top-Down-Prinzip entwickelt, um ein deterministisches Verhalten zu bestimmen (Rochner/Müller-Schloer 2005, 197). Im Gegensatz dazu folgen vollständig emergente Systeme in ihrer Entwicklung dem Bottom-Up-Verhalten, wobei eine konzise Bestimmung des Systemverhaltens nicht gegeben ist (Rochner/Müller-Schloer 2005, 197). Um die Entwicklung von technischen Systemen mit emergentem Verhalten zu un-

terstützen, ist ein Kompromiss zwischen diesen Entwicklungsansätzen erforderlich (Rochner/Müller-Schloer 2005, 197). Um den Ausgang der Berechnung in einem solchen, nach dem selbstorganisierenden Prinzip implementierten technischen System zu steuern, schlagen *Rochner/Müller-Schoer* (2005, 197-199) Observer/Controller-Architekturen (siehe Abbildung 3.3-1) vor, welche im Folgenden beschrieben werden:

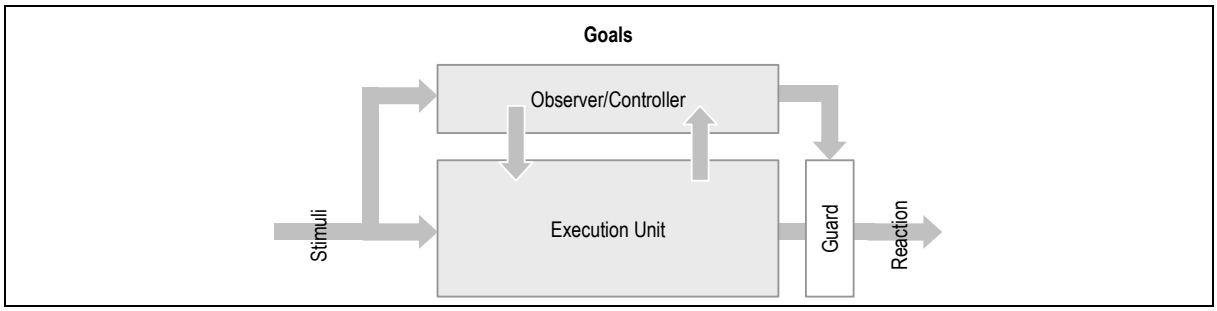

Abbildung 3.3-1: ***Observer/Controller-Architektur***
Quelle: *Rochner/Müller-Schloer* (2005, 197)

In einer solchen Architektur wird zwischen den Elementen Stimuli als Eingabe, Ziele (Goals), Observer/Controller, Ausführungseinheit (Execution Unit), Filter (Guard) und Reaktion (Reaction) als Ausgabe unterschieden. Sowohl die Ausführungseinheit als auch der Observer beobachten Veränderungen in der Umwelt, die sie als Stimuli wahrnehmen. Der Observer betrachtet weiterhin das Verhalten der Ausführungseinheit. Das sich dort entwickelnde Verhalten ist emergent. Werden im emergenten Verhalten der Ausführungseinheit ungewollte Entwicklungen identifiziert, wird der Controller vom Observer benachrichtigt. Der Controller vergleicht das Verhalten der Ausführungseinheit mit den vorab festgesetzten Zielen. Wird dabei eine Diskrepanz festgestellt, wird die Ausführungseinheit vom Controller entsprechend den globalen Zielen rekonfiguriert. Somit erreicht nur gewolltes emergentes Verhalten die Umwelt.

Mit dieser Architektur wird eine zusätzliche Abstraktionsschicht eingeführt, denn die Ziele des Systems werden auf abstraktem Niveau definiert, wobei es Aufgabe des Controllers ist, diese Ziele in Konfigurationen der Ausführungseinheit zu übersetzen (Rochner/Müller-Schloer 2005, 198). Die Formulierung von Zielen und die Übersetzung dieser Ziele in adäquate Veränderungen in der Ausführungseinheit sind bisher im Wesentlichen noch ungelöst (Rochner/Müller-Schloer 2005, 199). Den folgenden beiden Ansätzen wird dabei ein bedeutendes Potenzial eingeräumt (Rochner/Müller-Schloer 2005, 199): Zusicherungen oder der bereits erwähnte Filter. Zusicherungen (Meyer 2000) sind logische Ausdrücke, mit denen Programmtext annotiert wird. Darin werden z.B. logische oder temporale Bedingungen beschrieben (Rochner/Müller-Schloer 2005, 199): Sind diese nicht erfüllt, wird ein entsprechender, in der Bedingung formulierter Zweig ausgeführt. In dem Fall der Observer/Controller-Architektur kann dieser Zweig z.B. den Aufruf einer Methode im Observer bedeuten (Rochner/Müller-Schloer 2005, 199). Die zweite Möglichkeit des Filters leitet nur diejenigen Ausgaben der Ausführungseinheit weiter, die den Regeln genügen, die durch den Controller festgelegt sind (Rochner/Müller-Schloer 2005, 199).

Der Vorteil der beschriebenen Observer/Controller-Architektur kann der Übergang von einem vollständig emergenten System zu einem gesteuert emergenten System sein (Rochner/Müller-Schloer 2005, 199). Die Steuerung des Systems mit einer solchen Architektur liegt nicht ausschließlich beim Entwickler, der die globalen Ziele spezifiziert, sondern ein derart gesteuert emergentes System lässt bei der Umsetzung zur Laufzeit lokale Freiräume offen (Rochner/Müller-Schloer 2005, 199). Damit ist die Spezifikation des Verhaltens durch Ziele auf einem höheren Abstraktionsniveau möglich, als es heute noch bei technischen Systemen mit ihren vollständigen Spezifikationen der Fall ist. Selbstorganisierende Systeme tragen somit zur Beherrschung der Komplexität von technischen Systemen bei.

Den bisherigen Ausführungen zum Organic Computing folgend, kann der Zusammenhang mit Software-Agenten wie folgt hergestellt werden: In dem Agenten-Modell von *Wooldridge* (2002, 16) wird die Umwelt über Sensoren wahrgenommen. Diese Eingabe für einen Agenten entspricht den Stimuli, die eine Observer/Controller-Architektur erreichen. Das Ergebnis der Berechnungen innerhalb eines Agenten fließt über die durchgeführten Aktionen wieder zurück in die Umwelt. Dies entspricht der Reaktion einer Observer/Controller-Architektur. Somit besteht auch in dem Agentenmodell eine Relation zwischen Ein- und Ausgabe.

Bei Software-Agenten werden insbesondere im BDI-Ansatz (siehe dazu Abschnitt 2.3.1.4) Ziele definiert, die der Agent erreichen soll. *Pokahr/Braubach/Lamersdorf* (2005b) beschreiben dazu in ihrer Jadex-Architektur eine Deliberalisierungskomponente, welche entsprechend den gegebenen Ereignissen aus der verfügbaren Menge der Reaktionen oder Pläne ein geeignetes Element zur Ausführung aktiviert. Ereignisse werden dabei nicht nur von der Umwelt generiert, sondern auch über die internen Beliefs, Goals und Plans (Pokahr/Braubach/Lamersdorf 2005b, 152). Auf diesem Weg werden die Ziele letztendlich auf Aktivitäten abgebildet. Diese Aufgabe entspricht somit derjenigen des Controllers in der Observer/Controller-Architektur: In Abhängigkeit von den jeweiligen Ereignissen wird die Ausführungseinheit des Agenten rekonfiguriert, indem adäquate Pläne aktiviert werden. Die bei *Pokahr/Braubach/Lamersdorf* (2005b) dargestellte Architektur besitzt nicht nur die Charakteristika der Observer/Controller-Architektur, sondern ist letzterer sogar insofern überlegen, als Ziele neu hinzukommen und dynamisch ihren Zustand ändern können, indem sie in Abhängigkeit vom gegebenen Kontext einen eigenen Lebenszyklus durchlaufen (siehe Pokahr/Braubach/Lamersdorf 2005b, 153). Somit muss in der Architektur von *Pokahr/Braubach/Lamersdorf* (2005b) keine Instanz gegeben sein, die das emergente Verhalten beobachtet, wie es der Rolle des Observers entspricht.

Die ständige Konfiguration der Ausführungseinheit des Agenten erfolgt durch die adäquate Auswahl von Aktivitäten. Somit erreichen keine ungewollten Ausgaben die Umgebung. Deshalb ist eine Filterfunktionalität in der Agentenarchitektur nicht erforderlich, welche verhindern müsste, dass ungewollte Aktionen aktiviert werden.

Zusammenfassend kann festgestellt werden, dass Software-Agenten geeignete Komponenten sind, um Observer/Controller-Architekturen zur Realisierung von emergenten Systemen zu konstruieren. Die bei Pokahr/Braubach/Lamersdorf (2005b) beschriebene Architektur geht dabei sogar einen Schritt weiter, indem die Ziele einen eigenen Lebenszyklus durchlaufen und damit abhängig vom jeweiligen Kontext sind.

Software-Agenten kann für zukünftige Anwendungen im Sinne des proaktiven Computings ein wesentlicher Beitrag eingeräumt werden. Sowohl für die Realisierung der ambienten Intelligenz als auch des Organic Computing können Software-Agenten vorteilhaft eingesetzt werden (Kephart/Chess 2003, 45; Bauer/Odell 2005; Sterritt 2005, 83).

3.3.5 Agenten und Komponentenorientierung

Software-Agenten lassen sich zu Komponenten mit wohl definierten Schnittstellen in der Form von möglichen Interaktionen zusammenfassen. Gemäß den Ausführungen bei *Odell/Nodine/Levy* (2004) zur Klassifikation von Agentensubsystemen nach den Kriterien physikalische Eigenschaften, Rollen und Gruppen (siehe dazu Abschnitt 2.3.3) entspricht ein Subsystem aus Agenten nach außen einem einzigen Agenten (Jennings 2001, 39), der die durch seine internen Agenten angebotenen Dienste aggregiert und für andere Agenten anbietet. Die Dienstbeschreibung eines isolierten Agenten bzw. ihre Aggregation nach dem eben dargestellten Prinzip gibt damit die jeweilige Schnittstelle an. Somit können einzelne Agenten oder auch Agentensubsysteme als eine Komponente im Sinne der Komponentenorientierung (siehe dazu auch die Ausführungen in Abschnitt 2.3.1.2) betrachtet werden.

Der Ansatz bei *Brazier/Catholijn/Treur* (2002) geht mit diesen Ausführungen konform und erweitert diese, indem nicht nur das gesamte Multiagentensystem als eine Komponente betrachtet wird, aus der durch Dekomposition ihre Subsysteme bzw. Agenten erhalten werden (Brazier/Catholijn/Treur 2002, 3). Vielmehr werden dabei auch Agenten selbst in Komponenten zerlegt.

Der Kompositionskontext im Sinne der Komponentenorientierung kann bei Agenten durch ein Metamodell (Odell/Nodine/Levy 2004) bestimmt werden, welches die Bildung von dynamischen Agentensubsystemen determiniert, indem fehlende Dienste zur Laufzeit zu einer Agentenaggregation von einem Agenten mit geeigneter Dienstfunktionalität bereit gestellt werden. Außerdem zeigen *Bauer/Odell* (2005), dass der Kompositionskontext auch mit Mitteln der UML 2.0 über Kompositionsstruktur- und Komponentendiagramme beschrieben werden kann. Der im Zusammenhang mit der Komponentenorientierung relevante Deployment-Kontext für Agenten kann über Deployment-Diagramme der UML 2.0 spezifiziert werden (Bauer/Odell 2005).

Diese Ausführungen zeigen entsprechend den in Abschnitt 3.2.3.5 dargestellten Eigenschaften eines komponentenorientierten Systems, dass komponentenbasierte Architekturen auf der Basis von Software-Agenten umsetzbar sind. Voraussetzung dafür ist die Existenz von akzeptierten Standards für MASe und ihre Entwicklung. Da sich die Standardisierung hinsichtlich Modellierung, Kommunikation und Entwicklungsmethodiken jedoch aktuell noch im Wandel befindet, ist zunächst ihre breit akzeptierte Vereinheitlichung Voraussetzung für eine agentenbasierte Realisierung der Komponentenorientierung. Dies ist umso mehr relevant, als bei der Integration eines MAS in eine CORBA-Umgebung aktuell noch Schwächen identifiziert werden können (Pichler/Plösch/Weinreich 2002, 96-100). Im Gegensatz dazu stehen mit Java, CORBA und COM bzw. CLR (siehe dazu die Ausführungen in den Anhängen A.1 mit A.3) bereits etablierte Technologien der Komponentenorientierung zur Verfügung, für die jeweils verbreitete Implementierungen für umfassende Dienstfunktionalitäten existieren. Für eine breite Akzeptanz von Agenten als Möglichkeit zur Realisierung von Komponenten sind

dementsprechend eine geeignete Integration der genannten Technologien in ein MAS zu erarbeiten bzw. eine Bereitstellung von standardisierten und umfangreichen Diensten in Agentenplattformen erforderlich

Somit kann zusammenfassend festgestellt werden, dass die Komponentenorientierung auch auf der Basis von Software-Agenten realisiert und deren Vorteilhaftigkeit zur Steigerung des Abstraktionsniveaus angewendet werden kann. Dabei sind insbesondere die Besonderheiten der Flexibilität, Adaption und Selbstorganisation (Bauer/Odell 2005) von Interesse, um die Anforderungen von komplexen Software-Systemen zu erfüllen. Insgesamt werden die Vorteile aus der Komponentenorientierung und von agentenbasierten Systemen vereint. Voraussetzung für eine breite Akzeptanz der agentenbasierten Komponentenorientierung ist jedoch die Bereitstellung von durchgängigen Standards bzw. die Integration von bereits etablierten Komponententechnologien in Agentensysteme.

3.3.6 Agenten und EAI

Im Zusammenhang mit den Ausführungen zur EAI in Abschnitt 3.2.4.1 kann eine agentenbasierte Integrationsarchitektur im Sinne von *Strüver* (2006, 40-62) folgende Eigenschaften besitzen, durch welche die zur Umsetzung einer EAI erforderliche Konnektivität, Daten-, Objekt- sowie Prozessintegration in einem Agentensystem konkretisiert werden:

Die *Konnektivität* wird durch Kommunikationsmechanismen realisiert, welche von der jeweiligen Plattform zur Verfügung gestellt und vor dem Anwendungsentwickler verborgen werden. Dazu können, je nach gewählter Infrastruktur, unterschiedliche Technologien eingesetzt werden, von denen durch die Agentenplattform abstrahiert wird. Auf dieser Basis wird der Message Broker einer EAI in technischer Hinsicht in einer FIPA-Plattform durch die Komponente „Internal Platform Message Transport“ realisiert. Mit diesem Infrastrukturdienst in Form einer Bus Topologie werden einerseits die Elemente innerhalb einer Agentenplattform verbunden. Andererseits wird die Kommunikation mit Komponenten anderer Plattformen im Falle einer FIPA-Plattform durch den ACC erreicht. Somit können Bus Topologien zu vermaschten Netzwerken erweitert werden, deren Knoten jeweils eine Bus Topologie darstellen. Mit diesen Ausführungen wird deutlich, dass bei der Realisierung einer EAI über ein agentenbasiertes System zur Gewährleistung der Konnektivität notwendigerweise eine geeignete Agentenplattform bereitzustellen ist. Dabei ist aus Gründen der Interoperabilität zu beachten, dass diese mit etablierten Standards wie z.B. FIPA kompatibel ist. Obwohl bei der Verwendung einer MASIF-Plattform (siehe dazu Abschnitt 2.3.4.3) die Interoperabilität z.B. mit CORBA generell gegeben ist, identifizieren die Ausführungen bei *Pichler/Plösch/Weinreich* (2002, 96-100) Schwächen bei der Integration eines MAS in eine CORBA-Umgebung. Mit der Wahl einer Agentenplattform erfolgt insgesamt eine Festlegung auf eine einzige Infrastruktur, womit einerseits die Einschränkung auf diese gegeben ist, aber andererseits die Komplexität zum Management der Integrationstechnologie reduziert wird. Im Gegensatz dazu können bei einer EAI heterogene Integrationstechnologien eingesetzt werden. Damit wird eine höhere Flexibilität hinsichtlich der Implementierungstechnologie erreicht. Zu beachten ist dabei aber, dass damit auch eine zunehmende Komplexität des Message Brokers für das Management der Integrationstechnologien gegeben ist.

Die *Datenintegration* erfolgt in einem Agentensystem über das Konzept der Wrapper (Jennings 2001, 40). Diese wiederum müssen sich Mittel wie z.B. JDBC (zu JDBC als Ansatz zur Datenintegration in der EAI siehe Strüver 2006, 50) zunutze machen, um Daten aus Daten-

banken integrieren zu können. Dazu werden eingehende Agentennachrichten an den Wrapper in SQL-Anweisungen (Structured Query Language) transformiert und von der Datenbank verarbeitet. Die von dort erhaltenen Ergebnisse sind vor ihrem Versand vom Wrapper geeignet in eine Agentennachricht zu überführen.

Die *Objektintegration* erfolgt in einem Agentensystem einerseits auf der Basis der Kommunikation zwischen Agenten über ACL-Nachrichten, andererseits können in diesen Nachrichten auch vollständige Objekte in Form von Ontologie-Elementen ausgetauscht werden. Der Kommunikationsmechanismus zwischen den Agenten ist somit lose gekoppelt (siehe dazu Abschnitt 3.4.1).

Für die Ebene der *Prozessintegration* ist die Berücksichtigung der Ablauflogik relevant (Strüver 2006, 57). Für eine geeignete Integration der Systeme ist eine Trennung der Ablauflogik von der Implementierung vorteilhaft (Strüver 2006, 58). In einer agentenbasierten Realisierung ist deshalb ein geeigneter modularer Ansatz zu wählen, der die Integration unterschiedlicher Teilprozesse erlaubt. Die Flexibilität, die dabei in agentenbasierten Systemen gegeben ist (Jennings/Sycara/Wooldridge 1998, 15), erlaubt eine dynamische Anpassung der Prozesse entsprechend den Anforderungen.

Die Ausführungen in diesem Abschnitt zeigen, dass Software-Agenten im Sinne der EAI als Technologie zur Integration von Informationssystemen eingesetzt werden können. Dabei wird insbesondere durch die Charakteristika von Agenten die dynamische Anpassung (Sutherland/van den Heuvel 2002, 59 und 64) an die Umgebung ermöglicht. Für eine geeignete Realisierung einer EAI mit diesen Eigenschaften sind die Komponenten der Unternehmens-Software im Sinne von komplexen, adaptiven Systemen (CAS, für eine Beschreibung dieses Terminus siehe Abschnitt 5.1) zu „agentifizieren", was bei bisherigen Ansätzen jedoch nicht gegeben ist (Sutherland/van den Heuvel 2002, 64). Insbesondere erweist sich der EAI-Ansatz über Web Services für ex ante nicht vorgesehene Anfragen im Vergleich mit einer Implementierung auf der Basis von Agentensystemen als zu wenig flexibel (Kamel Boulos et al. 2006, 168). Bei letzteren ist die lose Kopplung (siehe dazu Abschnitt 3.4.1) von wesentlichem Vorteil. Somit stellt eine agentenbasierte Architektur zur Realisierung von EAI-Ansätzen einen geeigneten Mechanismus zur Verfügung, der zukünftigen Anforderungen der flexiblen Anpassung an sich ändernde Rahmenbedingungen gewachsen ist.

3.3.7 Agenten und SOA

Im Kontext der Eigenschaft einer (auch agentenbasierten) EAI (siehe dazu Abschnitt 3.3.6) als Treiber für die Etablierung einer SOA (Krafzig/Banke/Slama 2006, 215) wird im Folgenden dargestellt, wie eine SOA auf der Basis eines agentenbasierten Systems umgesetzt werden kann:

- Das *Applikations-Frontend* wird durch eine dedizierte Komponente mit Benutzerschnittstelle realisiert, die unterschiedliche Sichten für die jeweiligen Anwender zur Verfügung stellt und durch die die Aktionen der gesamten Applikation gesteuert werden. Dabei können etablierte Architekturmuster wie Schichten oder MVC eingesetzt werden.
- *Dienste* werden durch die nach außen sichtbaren Funktionalitäten einer Gruppe von Agenten oder von atomaren Agenten angeboten. Die Dienstperspektive kann somit auf unterschiedlichen Ebenen beobachtet werden: Auf unterster Ebene können einzelne Agenten sinnvolle Dienste anbieten (Lockemann/Nimis 2005, 30). Betrachtet man Agenten in einer Gruppe, die bestimmte Dienste anbieten (siehe dazu Jennings 2001; Odell/Nodine/Levy 2004), kann auch hier der Dienstbegriff angesetzt werden.

- Ein Dienst wird durch Funktionalitäten eines Agenten oder einer Gruppe von Agenten *implementiert*. *Daten* werden über das Wrapper-Konzept (Jennings 2001, 40) gekapselt. Die *Geschäftslogik* wird über die Dienstschnittstellen der Agenten verborgen.
- Der *Contract* für den von einem Agenten angebotenen Dienst ist durch seine Dienstbeschreibung gegeben. Diese wird im Sinne der SOA als *Interface* in einer FIPA-kompatiblen Plattform im DF gespeichert, um anderen Agenten die Nutzung des angebotenen Dienstes zu ermöglichen. Das *Dienstverzeichnis* ist in einem agentenbasierten System mit FIPA-Kompatibilität durch den DF gegeben.
- Der *Dienstbus* wird aus technischer Perspektive durch die Komponente „Internal Platform Message Transport“ realisiert.
- Die zu integrierenden *Applikationen* werden über dedizierte Wrapper-Agenten (Jennings 2001, 40) in das Agentensystem integriert, um dort über Dienste ihre Funktionalität zur Verfügung zu stellen.

Mit dieser Darstellung wird deutlich, dass eine SOA grundsätzlich durch ein MAS realisiert werden kann. Ein Dienst, der von einem Agenten zur Verfügung gestellt wird, besitzt aber nicht per se die Charakteristika, die von einem Dienst im Sinne der SOA erfüllt werden. Deshalb ist bei einer agentenbasierten Realisierung einer SOA zu gewährleisten, dass das Dienstspektrum der durch einen Agenten implementierten Funktionalität über die gekapselte Geschäftsentität oder -funktionalität definiert wird. Wird ein Dienst entsprechend den SOA-Eigenschaften agentenbasiert implementiert, lassen sich dadurch wesentliche Vorteile gegenüber einer herkömmlichen Realisierung umsetzen: Dienste können flexibel entsprechend den Anforderungen zur Verfügung gestellt werden. Dies kann, der Darstellung bei *Odell/Nodine/Levy* (2004) folgend, zur Laufzeit erfolgen, indem Agentengruppen dynamisch komponiert und wieder aufgelöst werden, um ihre Dienstleistungen zu erbringen. Weiterhin zeichnen sich Agentensysteme durch die redundante Bereitstellung von Diensten aus. Damit wird durch eine agentenbasierte Realisierung einer SOA noch mehr Flexibilität zur Reaktion auf sich dynamisch ändernde Rahmenbedingungen realisiert, als dies über eine herkömmliche Implementierungstechnik erzielt werden kann. Ein weiterer Differenzierungsaspekt liegt in der in einem Agentensystem gegebenen Autonomie. Während in einem solchen System die Ausführung der angeforderten Dienste der Entscheidung des jeweiligen Agenten unterliegt, werden Aufrufe in einer herkömmlichen Implementierung einer SOA meist ohne weitere Überprüfung ausgeführt. Darüber hinaus wird entsprechend der Definition eines von einem Agenten erbrachten Dienstes in Abschnitt 2.3.2 die Dienstqualität wesentlich durch die Eigenschaften der Reaktivität, Ausgewogenheit zwischen Reaktivität und Ziel-Orientierung, Proaktivität, soziale Fähigkeit sowie Lernfähigkeit bestimmt. Ergänzt werden können diese Eigenschaften durch die Emotionalität und Antagonistik eines Agenten. Diese Charakteristika sind bei einer Dienstdefinition im Sinne der SOA nicht gegeben.

Diese Ausführungen zeigen, dass der Dienstbegriff in einer SOA bzw. einem MAS nicht auf gleicher Ebene angesiedelt werden kann: Obwohl ein Dienst im Sinne der SOA durch eine agentenbasierte Implementierung zur Verfügung gestellt werden kann, besitzen Dienste in einem MAS die Eigenschaft der flexiblen und redundanten Komposition bzw. Dekomposition sowie die durch Agenten induzierte Autonomie. Darüber hinaus können von Agenten erbrachte Dienste durch weitere Eigenschaften charakterisiert werden, die über das Verständnis eines Dienstes im Sinne der SOA hinaus gehen.

Hinsichtlich Dienstbus ist ein wesentlicher Unterschied zwischen einer SOA und einem MAS erkennbar: Während in einer SOA über den Dienstbus heterogene Kommunikationsmechanismen und -technologien unterstützt werden, existiert in einem Agentensystem üblicherweise nur eine einzige Plattform. Mit der FIPA-Standardisierung ist zwar die Kommunikation zwischen FIPA kompatiblen Agentenplattformen gegeben, bei der Integration heterogener Plattformen und Middlewares wie z.B. CORBA können jedoch Schwächen identifiziert werden (Pichler/Plösch/Weinreich 2002, 96-100). Damit ist eine SOA auf der Basis eines MAS hinsichtlich der zur Verfügung stehenden und etablierten Integrationstechnologien eingeschränkt. Während damit zwar die bei einer SOA gewünschte Flexibilität der eingesetzten Technologien beschränkt ist, wird die Komplexität der SOA-Implementierung reduziert, weil nur eine einzige Technologie fokussiert wird.

Zusammenfassend kann konstatiert werden, dass mit diesen Ausführungen eine SOA auch über ein agentenbasiertes System realisiert werden kann. Insbesondere besitzen agentenbasierte Systeme die Eigenschaft der dynamischen und redundanten Komposition bzw. Dekomposition von Diensten. Deshalb eignen sie sich vorteilhaft für eine flexible Anpassung an sich verändernde Rahmenbedingungen, die auch zur Laufzeit beherrscht werden können. Dazu trägt auch die in einem Agenten mögliche Autonomie bei, die über die Durchführung des gewünschten Dienstes entscheidet. Weiterhin kann ein durch einen Agenten erbrachter Dienst durch die Eigenschaften der Reaktivität, Ausgewogenheit zwischen Reaktivität und Zielorientierung, Proaktivität, sozialen Fähigkeit, Lernfähigkeit, Emotionalität und Antagonistik charakterisiert werden. Nachteilig bei einer agentenbasierten Realisierung einer SOA ist jedoch die Einschränkung auf die gewählte Agentenplattform bzw. den dadurch induzierten Standard. Dies ist bei der Implementierung einer SOA über herkömmliche Technologien nicht gegeben. Insgesamt sind bei einer Entscheidung über die verwendete Technologie die jeweiligen Vor- und Nachteile gegeneinander abzuwägen.

3.3.8 Agenten und Web Services

Software-Agenten und Web Services besitzen als offensichtliche Gemeinsamkeiten den Dienstbegriff sowie die Anforderung einer geeigneten Infrastruktur zur Registrierung, Suche und Nutzung von Diensten. Damit sind wesentliche Voraussetzungen für eine Integration der beiden Technologien gegeben. Das dafür eingesetzte Verfahren wird in Abschnitt 6.3.5 beschrieben, mit dem Web Services in eine Agentenumgebung eingebettet werden können.

Ein wesentliches Differenzierungsmerkmal von Software-Agenten gegenüber Web Services ist in der Autonomie von Agenten gegeben. In Analogie zum Vergleich zwischen der Objektorientierung und Software-Agenten können letztere über die Ausführung von gewünschten Diensten selbst entscheiden. Bei Objekten bzw. Web Services ist diese Entscheidungskompetenz nicht gegeben.

Zusammenfassend lässt sich somit feststellen, dass Web Services und Software-Agenten wegen ihres gemeinsamen Dienstmerkmals kombiniert werden können. Es ist allerdings festzuhalten, dass Software-Agenten im Vergleich zu Web Services autonom darüber entscheiden können, ob der angeforderte Dienst ausgeführt wird oder nicht.

3.4 Evaluierung von Architekturen

Die bisherigen Ausführungen zu Architekturen bilden die Grundlage für ihre Evaluierung in diesem Abschnitt.

Entsprechend dem Modell des Informationsmanagements bei *Krcmar* (2005, 47-49), welches „das Management der Informationswirtschaft, der Informationssysteme, der Informations-

und Kommunikationstechniken sowie der übergreifenden Führungsaufgaben“ (Krcmar 2005, 49) umfasst, wird im Folgenden die Evaluierung auf den Ebenen der Informationssysteme sowie der Informations- und Kommunikationstechnik (für eine geeignete Fokussierung auf Krankenhausinformationssysteme siehe Winter et al. 1999; Winter et al. 2001) angestrebt. Mit diesen beiden Ebenen werden in der vorliegenden Arbeit die gewünschten Grundlagen geschaffen, um das Management der Informationswirtschaft geeignet unterstützen zu können. Letzteres wird in der vorliegenden Arbeit jedoch nicht berücksichtigt.

Informationssystemarchitekturen (IS-Architekturen, Ebene der Informationssysteme im Modell bei Krcmar 2005, 47) werden bei *Krcmar* (2005, 195-196) entlang der folgenden Kriterien klassifiziert:

- *Offen*: Konstruktion der IS-Architektur aus Komponenten mit Schnittstellen, die offene Standards unterstützen
- *Integriert*: Komposition der IS-Architektur aus Komponenten mit proprietären Schnittstellen
- *Verteilt*: Institutionsübergreifende Komposition
- *Geschlossen*: Komposition nur innerhalb einer Institution

Entsprechend dieser Klassifikation werden in der vorliegenden Arbeit zur Evaluierung diejenigen IS-Architekturen betrachtet, die auf der Basis der beschriebenen Software-Architekturen konstruiert werden können. Damit werden letztere hinsichtlich ihrer Potenziale zur Realisierung von IS-Architekturen bewertet.

Um die bisher dargestellten Software-Architekturen (Ebene der Informations- und Kommunikationstechnik bei Krcmar 2005, 47) zueinander in Relation setzen und damit ihre Eignung zur Erfüllung von Anforderungen systematisch aufzeigen zu können, werden in Anlehnung an das Vorgehen bei *Stoermer/Bachmann/Verhoef* (2003) aus den Anforderungen für Informationssysteme zur Integration relevante Kriterien einer geeigneten Software-Architektur abgeleitet und diese jeweils hinsichtlich ihres Deckungsgrades bewertet. Ein solcher Vergleich auf der Basis von Architekturen ist gerechtfertigt, weil diese die Voraussetzung für die Implementierung eines Systems entlang relevanter Qualitätskriterien bilden (Bass/Clements/Kazman 2003, 73; de Bruin/van Vliet 2003) und diese Kriterien damit die Ausprägung einer Architektur bestimmen (Wijnstra 2003, 199). Solche Kriterien umfassen oftmals die folgenden allgemeinen Aspekte[13]:

- Modifizierbarkeit: Skalierbarkeit, Portierbarkeit, Wartbarkeit, Konfigurierbarkeit, Erweiterungsfähigkeit
- Performanz: Durchsatz, Antwortzeit
- Sicherheit: Datensicherheit, -schutz, Verfügbarkeit, Zuverlässigkeit, Fehlertoleranz
- Testbarkeit
- Benutzerfreundlichkeit

[13] *Lundberg et al.* (1999), *Bass/Clements/Kazman* (2003, 71-98), *Bosch/Lundberg* (2003, 183 und 185), *Wijnstra* (2003, 201-202).

- Interoperabilität, Integration von Altsystemen
- Wiederverwendbarkeit
- Flexibilität

Diese Qualitätsattribute gelten für beliebige Software-Architekturen. Weil in der vorliegenden Arbeit agentenbasierte Systeme fokussiert werden und durch ein MAS eine spezielle Architektur gegeben ist (Shehory 2000, 77), ist auch die Berücksichtigung der besonderen Architektureigenschaften von MASen erforderlich (Shehory 2000, 77 und 79-84 sowie 89; Davidsson/Johansson 2002):

- Robustheit
- Adaptierbarkeit
- Systemoffenheit
- Lastverteilung
- Skalierbarkeit

Die genannten Qualitätsattribute sind jeweils für die angestrebten Anforderungen auf ihre Relevanz zu überprüfen und ggf. durch weitere Aspekte zu ergänzen. Die Erhebung dieser Anforderungen zur Integration von Informationssystemen wird im folgenden Abschnitt vorgenommen.

3.4.1 Evaluierungskonzept

Als Basis für den Architekturvergleich wird in diesem Abschnitt ein Evaluierungskonzept erarbeitet, durch das die Anforderungen eines Informationssystems zur Integration reflektiert werden.

Auf IS-Architekturebene eignen sich offene und verteilte Architekturen, um die Integration über Institutionsgrenzen hinweg zu unterstützen. Deshalb wird zur Evaluierung der Software-Architekturen in der vorliegenden Arbeit beleuchtet, ob auf deren Basis offene und verteilte IS-Architekturen konstruiert werden können.

Die Integration von Informationssystemen, insbesondere über Organisationsgrenzen hinweg, bedarf der besonderen Unterstützung hinsichtlich der variierenden Anforderungen und Rahmenbedingungen. Weiterhin ist die Unterstützung für eine Vielzahl unterschiedlicher Systeme und Standards zu leisten. Damit sind vorrangig die aus Abschnitt 3.4 gewählten Qualitätsattribute zu gewährleisten:

- Modifizierbarkeit
- Interoperabilität
- Flexibilität
- Systemoffenheit

Auf technischer Ebene sind dazu insbesondere eine geeignete Modularisierung sowie adäquate Kopplung zwischen den Architekturelementen zu unterstützen, um die genannten Kriterien

geeignet zu erfüllen. Konzepte der Modularisierung wurden bereits in den Abschnitten zur Objektorientierung (Abschnitt 2.2.5) sowie zu grundlegenden Architekturtypen (Abschnitt 3.2.3) und Architekturen zur Integration beschrieben. Deshalb wird auf eine erneute Ausführung in diesem Abschnitt verzichtet.

Zur Realisierung von Kommunikationsmechanismen zwischen Architekturkomponenten wird zwischen enger und loser Kopplung differenziert. Die Bedeutung dieses Terminus ist abhängig von unterschiedlichen Betrachtungsebenen. Diese sind in Tabelle 3.4-1 zusammengefasst und werden im Folgenden detailliert beschrieben (siehe auch Krafzig/Banke/Slama 2006, 38-49):

Ebene	Enge Kopplung	Lose Kopplung
Physikalische Kopplung	Direkte physikalische Verbindung erforderlich	Physikalischer Mediator
Kommunikationsstil	Synchron	Asynchron
Typsystem	Starkes Typsystem (z.B. Schnittstellensemantik)	Schwaches Typsystem (z.B. Semantik des Nachrichteninhalts)
Interaktionsmuster	Objektorientierter Stil zur Navigation von komplexen Objektbäumen	Datenzentrierte, selbst beinhaltete Nachrichten
Steuerung der Prozesslogik	Zentrale Steuerung der Prozesslogik	Verteilte logische Komponenten
Dienstsuche und -bindung	Statisch gebundene Dienste	Dynamisch gebundene Dienste
Plattformabhängigkeiten	Starke Betriebssystem- und Programmiersprachenabhängigkeiten	Unabhängigkeit von Betriebssystemen und Programmiersprachen

Tabelle 3.4-1: ***Enge und lose Kopplung in Abhängigkeit von der jeweils betrachteten Ebene***
Quelle: In Anlehnung an *Krafzig/Banke/Slama* (2006, 47)

Hinsichtlich des Aspektes der *physikalischen Kopplung* kommunizieren eng gekoppelte Systeme über eine direkte Verbindung z.B. über die Dauer des entfernten Prozeduraufrufs. Bei einer losen Kopplung werden dagegen die Nachrichten zwischen Sender und Empfänger über eine Nachrichtenwarteschlage ausgetauscht. Damit wird die Kommunikation über einen physikalischen Mediator realisiert und entkoppelt.

Der *Kommunikationsstil* ist abhängig von der physikalischen Kopplung. Synchrone Kommunikation basiert auf einer direkten Verbindung zwischen Sender und Empfänger, bei asynchroner Kommunikation werden die beiden Kommunikationspartner z.B. über eine Warteschlange entkoppelt.

Bei enger Kopplung kann ein starkes *Typsystem* festgestellt werden. Zum Aufruf einer Funktion müssen ihre Signatur, ihr Verhalten und ihr Behandlung von Ausnahmen bekannt sein (van Aart et al. 2002). Dies wird als Schnittstellensemantik von Funktionsaufrufen bezeichnet. Von Vorteil bei diesem Ansatz ist die Identifikation von Typfehlern bereits bei der

Systementwicklung, weil diese zum Zeitpunkt der Kompilierung identifiziert werden können. Werden die Inhalte hingegen über Nachrichten versendet, ist die Kopplung hinsichtlich des Typsystems lose. Dies wird mit schwachem Typsystem bzw. der Semantik des Nachrichteninhalts bezeichnet. Bei der losen Kopplung verlagert sich das Problem des Austausches von Nachrichten nun von dem Zeitpunkt der Kompilierung auf denjenigen der Laufzeit. Werden die Nachrichteninhalte verändert, treten erst zur Laufzeit Spezifikationsfehler der Nachrichten auf, wenn die Nachrichten nicht korrekt entpackt oder zugeordnet werden können.

Auf der Ebene der *Interaktionsmuster* ist nicht nur die Kenntnis der Objektsemantik erforderlich, sondern auch das Wissen über die Navigation in den Objektgeflechten. Damit ist eine enge Bindung zwischen Client und Server gegeben. Bei der losen Kopplung ist lediglich eine Abhängigkeit auf der Ebene des ausgetauschten Nachrichteninhalts erforderlich.

Bei enger Kopplung existiert eine zentrale *Steuerung der Prozesslogik*, bei loser Kopplung hingegen ist die Kontrolle verteilt auf die logischen Komponenten.

In einem eng gekoppelten System sind die *Dienste* statisch gebunden. Im Gegensatz dazu werden in lose gekoppelten Systemen die Dienste durch die Suche in einem Namensdienst dynamisch gebunden.

Abhängigkeiten vom jeweils eingesetzten Betriebssystem oder den Programmiersprachen zeichnen ein eng gekoppeltes System aus. Lose gekoppelte Systeme sind dagegen unabhängig vom gewählten Betriebssystem oder der Programmiersprache.

Auf der Ebene der IS-Architekturen eignen sich für Integrationsszenarien offene und verteilte Architekturen, weil über die Komposition von Komponenten mit standardisierten Schnittstellen und über Institutionsgrenzen hinaus Flexibilität erreicht werden kann.

Je mehr Flexibilität bei der Interoperabilität benötigt wird, desto besser eignen sich auf der Ebene der Informations- und Kommunikationstechnik lose gekoppelte Systeme (Krafzig/Banke/Slama 2006, 49). Solche Systeme sind dann von Vorteil, wenn sie häufig angepasst werden müssen (Humm/Voß/Hess 2006, 404; Krafzig/Banke/Slama 2006, 49). Weiterhin kann über eine lose Kopplung die Robustheit gesteigert werden, wenn einzelne Komponenten aktuell nicht verfügbar sind (Humm/Voß/Hess 2006, 406). Trotzdem sind mit der losen Kopplung wesentliche Nachteile gegeben: Zusätzlicher „Aufwand für Integrationstechnik, Fehlerbehandlung und Sicherheit sowie schlechtere Performance“ (Humm/Voß/Hess 2006, 404-405). Deswegen werden neben den beschriebenen Extremen der losen und engen Kopplung auch Mischformen beobachtet. Je nach den gegebenen Anforderungen ist ein Kompromiss zwischen der Komplexität und dem Freiheitsgrad für eine flexible Anpassung an sich ändernde Rahmenbedingungen zu schließen (Humm/Voß/Hess 2006, 405). Mögliche Handlungsempfehlungen für diesen Entscheidungsprozess werden bei Humm/Voß/Hess (2006, 405-407) beschrieben, sind jedoch nicht Gegenstand der vorliegenden Arbeit.

3.4.2 Evaluierung

Auf der Grundlage des in Abschnitt 3.4.1 erarbeiteten Evaluierungskonzeptes mit den dort genannten Eigenschaften von IS-Architekturen sowie Qualitätsattributen von Software-Architekturen wird im Folgenden die Evaluierung der in den Abschnitten 3.2.3 und 3.2.4 beschriebenen Software-Architekturen vorgenommen. Dabei ist zu beachten, dass diese Architekturen nicht aus der in Abschnitt 3.3 beschriebenen Perspektive agentenbasierter Systeme betrachtet werden. Die Ergebnisse der Architekturevaluierung sind in Tabelle 3.4-2 zusammengefasst.

Kriterium / Architektur	Potenzielle IS-Architekturen	Maßnahmen zur Modularisierung	Kopplung
Client-Server-Architektur	• Abhängig von der Implementierung offen oder integriert • Abhängig von der lokalen Restriktion verteilt oder geschlossen	• Strikte Rollenverteilung resultiert in einer eingeschränkten Skalierbarkeit • Integration von Altsystemen über (standardisierte) Middlewares, eingeschränkt über die Implementierungskomplexität • Geringe Flexibilität durch enge Kopplung • Eingeschränkte Systemoffenheit	• Synchroner Kommunikationsstil • Schnittstellensemantik, starkes Typsystem • Steuerung der Prozesslogik durch Clients • Statisch gebundene Dienste • Programmiersprachen- und Betriebssystemabhängigkeit
Peer-to-Peer-Architektur	• Integriert • Verteilt	• Strukturierung durch leichtgewichtige Peers, jeweils ggf. in einer Doppelrolle als Client und Server, führt zu Skalierbarkeit durch Lastverteilung • Interoperabilität eingeschränkt durch die Vielzahl heterogener und nicht kompatibler Protokolle, die Heterogenität erfordert die Konsolidierung der Interaktionsfunktionalitäten oder den Einsatz von Proxy basierten Architekturen • Flexibilität durch Doppelrollen, redundante Bereitstellung von Funktionalitäten • Systemoffenheit durch Spezifikation von benötigten Ressourcen über Inhalte	• Synchroner oder asynchroner Kommunikationsstil in Abhängigkeit der Implementierung • Enge Kopplung über den typisierten Aufruf von Operationen • Verteilte Steuerung der Prozesslogik • Suche nach Ressourcen über den Inhalt, dynamische Dienstsuche und -bindung • Programmiersprachen- und Betriebssystemunabhängigkeit
Komponenten-orientierte Architektur	• Abhängig von der Implementierung offen oder integriert • Abhängig von der lokalen Restriktion verteilt oder geschlossen	• Modifizierbarkeit durch Erweiterbarkeit und Konfigurierbarkeit • Interoperabilität innerhalb der gleichen Technologie gegeben, Interoperabilität eingeschränkt zwischen heterogenen Technologien • Funktionalitäten werden von Komponenten angeboten bzw. sind für die Erbringung der Dienstleistung von anderen Komponenten zur Verfügung zu stellen, Rolle von Client und Server in einer Komponente vereint • Eingeschränkte Systemoffenheit	• Synchroner Kommunikationsstil • Enge Kopplung über die genaue Spezifikation des Installationskontextes, Schnittstellensemantik • Zentrale Steuerung der Prozesslogik • Dynamische Dienstsuche und -bindung • Programmiersprachen- und Betriebssystemunabhängigkeit bei entsprechender Middleware
EAI	• Abhängig von der Implementierung offen oder integriert • Abhängig von der lokalen Restriktion meist geschlossen, aber auch verteilt	• Geeignete Topologie des Message Brokers ermöglicht Erweiterbarkeit • Technologien zur Integration von Daten, Objekten und Prozessen aus den Informationssystemen • Flexibilität durch Doppelrollen • Eingeschränkte Systemoffenheit	• Synchroner oder asynchroner Kommunikationsstil in Abhängigkeit der Middleware • Enge Kopplung durch Schnittstellensemantik • Zentrale Steuerung der Prozesslogik • Dynamische Dienstsuche und -bindung bei entsprechender Middleware • Programmiersprachen- und Betriebssystemunabhängigkeit bei entsprechender Middleware

SOA	• Meist offen • Verteilt	• Komposition von an Geschäftsentitäten orientierten Diensten ermöglicht die Erweiterbarkeit • Heterogene Technologien zur Integration werden im Dienstbus vereint • Flexible Dienstkomposition • Eingeschränkte Systemoffenheit	• Synchroner oder asynchroner Kommunikationsstil in Abhängigkeit der Middleware • Enge Kopplung durch Schnittstellensemantik • Zentrale Steuerung der Prozesslogik • Dynamische Dienstsuche und -bindung bei entsprechender Middleware • Programmiersprachen- und Betriebssystemunabhängigkeit bei entsprechender Middleware
Web Services	• Offen • Verteilt	• Modifizierbarkeit durch Austauschbarkeit der Dienste mit wohl definierter Schnittstelle und dynamische Lastverteilung • Eingeschränkte Interoperabilität durch den verwendeten Protokoll-Stack • Flexibilität durch dynamische Bindung der Dienste, eingeschränkt durch die starre Architektur • Eingeschränkte Systemoffenheit	• Synchroner oder asynchroner Kommunikationsstil in Abhängigkeit der Implementierung • Eingeschränkte lose Kopplung über den Austausch von Inhalten über Nachrichten • Zentrale Steuerung der Prozesslogik • Dynamische Dienstsuche und -bindung • Programmiersprachen- und Betriebssystemunabhängigkeit
Agentenorientierte Architektur	• Offen • Verteilt	• Bedarfsorientierte und dynamische Komposition sowie Dekomposition von Subsystemen aus Agenten • Interoperabilität eingeschränkt durch Fokussierung auf eine ausgewählte Agentenplattform, Integration von Agentenplattformen in Middlewares wie CORBA bisher schwierig zu realisieren • Mögliche Doppelrolle eines Clients und Servers, wechselnde und mehrere Rollen zum gleichen Zeitpunkt, dynamische Lastverteilung • Vollständige Systemoffenheit	• Asynchrone Kommunikation • Lose Kopplung über den Austausch von in ACL-Nachrichten gekapselten Inhalten • Schwaches Typsystem • Dezentrale Steuerung der Prozesslogik • Dynamische Dienstsuche und -bindung • Eingeschränkte Programmiersprachen- und Betriebssystemunabhängigkeit

Tabelle 3.4-2: ***Ergebnisse der Evaluierung der Architekturen***
Quelle: Eigene Darstellung

Die Bewertung der Schichtenarchitektur ist nicht Gegenstand dieser Betrachtung, weil es sich dabei um eine Metaarchitektur handelt, die unabhängig von der gewählten Technologie realisierbar ist und deshalb keine signifikanten Unterschiede bei der Klassifizierung identifiziert werden können. Da die in Abschnitt 3.2.3.2 beschriebene verteilte Architektur nur generischen Charakter besitzt, der in den Ausprägungen der Architekturen in den Abschnitten 3.2.3.3 und 3.2.3.4 konkretisiert wird, wird in diesem Abschnitt auf ihre explizite Berücksichtigung verzichtet.

3.4.2.1 Evaluierung von Client-Server-Architekturen

Hinsichtlich der mit einer Client-Server-Architektur konstruierbaren IS-Architektur lässt sich feststellen, dass sich in Abhängigkeit von der Implementierung ein offenes oder integriertes System komponieren lässt. In Abhängigkeit von der lokalen Restriktion kann die IS-Architektur verteilt oder geschlossen gestaltet sein. Die durch eine Client-Server-Architektur induzierte IS-Architektur ist somit abhängig von den jeweiligen Rahmenbedingungen.

Client-Server-Architekturen zeichnen sich durch eine Modularisierung in Form einer strikten Aufteilung der Rollen in Client und Server aus. Damit ist die Modifizierbarkeit eines solchen Systems eingeschränkt: Der zentrale Server wird oftmals zu einem Flaschenhals, wodurch die Skalierbarkeit nur begrenzt möglich ist.

Die Interoperabilität mit anderen Systemen bzw. die Integration von Altsystemen kann über den Einsatz von Middleware erreicht werden. Voraussetzung ist dabei die Unterstützung von geeigneten Schnittstellen durch das Altsystem. Bei der Berücksichtigung einer Vielzahl von unterschiedlichen Middlewares steigt jedoch die Implementierungskomplexität durch die erforderliche Kopplung und Transformation auf den Ebenen der Kommunikationsformen und -inhalte. Deshalb wird die Integration von heterogenen Systemen durch die resultierende Implementierungskomplexität beschränkt. Vorteilhaft kann die bereits gegebene Spezifikation von etablierten und offenen Middlewares wirken, die die Interoperabilität zwischen heterogenen Systemen ermöglicht.

Mit einem Client-Server-System ist zwar eine verteilte Architektur und mit ihr die Reduktion der Komplexität durch die Rollenaufteilung gegeben, dennoch sind Client und Server meist so eng gekoppelt, dass die gesamte Anwendung nahezu als eine monolithische Implementierung betrachtet werden kann. Der Kommunikationsstil ist meist synchron. Die Kopplung auf Typsystemebene kann oftmals mit der Schnittstellensemantik charakterisiert werden. Die Steuerung der Prozesslogik erfolgt durch die Clients. Die Dienstsuche und -bindung finden zur Entwicklungszeit statt. Mit der Schnittstellensemantik geht eine enge Kopplung auf der Plattformebene einher. Zusammenfassend kann festgestellt werden, dass die Kopplung in einem Client-Server-System eng ist. Damit wird bei solchen Architekturen die Flexibilität zur Integration neuer Dienste nicht unterstützt (Steinmetz/Wehrle 2004, 51).

Eine zur Laufzeit bestehende Systemoffenheit ist in einer Client-Server-Architektur nur eingeschränkt gegeben, denn die von Servern angebotenen Dienste müssen bereits zum Zeitpunkt der Applikationskonstruktion spezifiziert sein, damit diese Clients zur Verfügung gestellt werden können.

Die IS-Architekturen, die auf der Grundlage von Client-Server-Architekturen konstruiert werden können, sind hinsichtlich ihrer Offenheit und Verteiltheit abhängig von der Art der Implementierung. Auf dieser Ebene kann somit, falls erforderlich, die gewünschte Flexibilität erzielt werden.

Insgesamt lässt sich hinsichtlich der Software-Architektur feststellen, dass in einem Client-Server-System Maßnahmen zur Modularisierung zwar gegeben, aber eingeschränkt sind. Weiterhin resultieren die Eigenschaften eines solchen Systems in einer relativ engen Kopplung, womit die Flexibilität nur bedingt möglich ist.

Zusammenfassend ist deshalb zu konstatieren, dass die für die Integration gewünschte Flexibilität auf der Ebene der IS-Architekturen zwar erreicht werden kann, aber durch die auf der Ebene der Software-Architektur gegebenen Restriktionen beschränkt wird.

3.4.2.2 Evaluierung von Peer-to-Peer-Architekturen

Werden IS-Architekturen auf der Basis eines Peer-to-Peer-Systems aufgebaut, sind diese meist integriert und verteilt. Die Eigenschaft der Integration ist bedingt durch die Festlegung auf bestimmte Standards zur Interaktion. Obwohl zur Lösung dieser Problematik bereits ein Ansatz existiert (siehe dazu weiter unten in diesem Abschnitt), kann die Konstruktion eines vollständig offenen Systems bisher nicht erreicht werden. Peer-to-Peer-Systeme sind per se verteilt und erlauben somit die Informationsintegration über Institutionsgrenzen hinaus. Insgesamt lässt sich feststellen, dass eine IS-Architektur auf der Basis eines Peer-to-Peer-Systems wegen ihrer integrierten Eigenschaften nur eingeschränkt flexibel ist.

Die Reduktion der Komplexität in Peer-to-Peer-Architekturen wird durch die Aufteilung in einzelne leichtgewichtige Peers erreicht. Die Lastverteilung kann dynamisch erfolgen, indem zur Laufzeit diejenigen Peers für eine Dienstleistung angefragt werden, die geeignete Ressourcen zur Verfügung stellen können. Die erforderlichen Lastanpassungen können in einem Peer-to-Peer-Netzwerk selbstorganisierend vorgenommen werden. Wesentlich ist dabei die Eigenschaft, dass die Kommunikation zwischen den Peers ohne jegliche mediierende Funktionalität eines Servers erfolgen kann, womit kein Flaschenhals entsteht. Außerdem erhöht die redundante Dienstbereitstellung durch mehrere Peers die Verfügbarkeit der Dienste und damit der gesamten Systemfunktionalität. Weiterhin können Peers jeweils neben einer Einzelrolle auch Doppelrollen einnehmen. Insgesamt kann mit diesen Ausführungen einer Peer-to-Peer-Architektur die Eigenschaft der Flexibilität attribuiert werden

Die Interoperabilität in einem Peer-to-Peer-Netzwerk kann durch die Verwendung eines offenen und einheitlichen Standards auf den Ebenen Kommunikationsprotokoll, Schnittstelle, Anfragesprache sowie Metadatenprofil zur Beschreibung der Semantik (Brunkhorst/Olmedilla 2006, 47-48) ermöglicht oder durch eine Proxy basierte Architektur (Brunkhorst/Olmedilla 2006) erreicht werden. Für die Realisierung der Interoperabilität über einen einheitlichen Standard existiert jedoch eine sehr umfangreiche Anzahl von heterogenen und nicht kompatiblen Ansätzen, die die Interoperabilität erschweren. Deshalb müssten die existierenden Standards für eine weite Verbreitung und einen Einsatz zur Implementierung von Informationssystemen geeignet vereinheitlicht werden. Wegen des Umfangs der gegebenen Standards erscheint dieses Vorgehen jedoch nicht durchführbar (Brunkhorst/Olmedilla 2006, 46). Deshalb schlagen *Brunkhorst/Olmedilla* (2006) einen Proxy basierten Ansatz vor: Dabei übernimmt der Proxy die Aufgabe der Transformation der Standards auf den genannten Ebenen und erlaubt somit die Interoperabilität heterogener Peer-to-Peer- und anderer Netze (Brunkhorst/Olmedilla 2006, 45). Obwohl dieser Ansatz vielversprechend ist, sind dennoch weitere Arbeiten erforderlich (Brunkhorst/Olmedilla 2006, 58). Insbesondere ist in diesem Zusammenhang anzumerken, dass die semantische Transformation nicht vollständig automatisierbar ist (Rahm/Bernstein 2001, 337). Deshalb bleibt die Interoperabilität zwischen Peer-to-Peer-Netzen auch bei dem Einsatz einer Proxy basierten Architektur eingeschränkt.

Die Kommunikation zwischen den Peers erfolgt, abhängig von der Implementierung, in einer asynchronen oder synchronen Weise, womit die Kopplung auf der Ebene des Kommunikationsstils lose bzw. eng gestaltet ist. Der Aufruf von Diensten wird über eine spezifizierte Schnittstelle gewährleistet. Damit werden die Peers über die Schnittstellensemantik eng gekoppelt. Bei der Suche nach benötigten Ressourcen in einem Peer-to-Peer-System können verteilte Hash-Tabellen eingesetzt werden. Damit erfolgt die Steuerung der Prozesslogik in den dezentralen Knoten. Die Suche nach benötigten Diensten wird auf der Basis von Inhalten durchgeführt. Damit wird von tatsächlichen Netzwerkadressen abstrahiert. Folglich können neue Teilnehmer dem Peer-to-Peer-Netz ohne Neukonfiguration des Systems zur Laufzeit hinzugefügt oder von ihm entfernt werden. Die Dienstsuche und -bindung erfolgen somit dynamisch. Weiterhin kann einem solchen System eine dynamische Systemoffenheit attribuiert werden. Weil die Suche nach Diensten nicht abhängig von der Adresse ist, können auf Plattformebene unterschiedliche Betriebssysteme und Programmiersprachen eingesetzt werden. Folglich kann auf dieser Ebene eine lose Kopplung identifiziert werden.

Auf der Ebene der durch Peer-to-Peer-Systeme konstruierten IS-Architekturen ist die Flexibilität nur eingeschränkt gegeben: Die IS-Architektur ist zwar per se verteilt, wegen der Vielzahl der gegebenen Standards aber integriert.

Bezüglich Software-Architektur sind die Maßnahmen zur Modularisierung zusammenfassend ausgeprägter, als sie in Client-Server-Architekturen realisiert sind. Es lässt sich weiterhin feststellen, dass eine lose Kopplung zwar unterstützt wird, diese aber eingeschränkt auf gewisse Ebenen ist. In ihrem Umfang ist die Kopplung im Vergleich zu Client-Server-Architekturen loser. Aus diesen Gründen sind Peer-to-Peer-Architekturen zur Integration von Informationssystemen zwar grundsätzlich, aber nur bedingt geeignet: Für eine Akzeptanz dieser Technologie sind geeignete Standards oder Proxy-Mechanismen zur Gewährleistung der Interoperabilität zu etablieren.

Zusammenfassend ist das Potenzial von Peer-to-Peer-Architekturen für einen Einsatz in Integrationsszenarien zwar gegeben, aber wegen der zu wenig ausgeprägten Flexibilität auf den Ebenen der IS- und Software-Architektur nur eingeschränkt umsetzbar.

3.4.2.3 Evaluierung von komponentenorientierten Architekturen

In IS-Architekturen, die auf der Basis von Software-Komponenten konstruiert werden, ist zwar die Interoperabilität durch den Einsatz von etablierten Technologien grundsätzlich gegeben. Die Interoperabilität zwischen heterogenen Technologien ist aber eingeschränkt (siehe dazu die Ausführungen weiter unten in diesem Abschnitt). Deshalb können komponierte IS-Architekturen offen oder integriert gestaltet sein. Je nach vorgegebener, lokaler Restriktion sind IS-Architekturen verteilt oder geschlossen. Insgesamt kann festgestellt werden, dass die bei IS-Architekturen auf der Basis einer komponentenorientierten Architektur erreichbare Flexibilität abhängig ist von der gewählten Implementierung und den lokalen Restriktionen.

In einer komponentenorientierten Software-Architektur wird die Komplexität beherrscht, indem die Anwendung aus einzelnen Komponenten komponiert wird. Diese Komponenten mit ihren wohl definierten Schnittstellen können entsprechend den Anforderungen zusammengestellt oder durch alternative Implementierungen ersetzt werden. Somit ist die Modifizierbarkeit eines komponentenorientierten Systems im Sinne der Erweiterbarkeit und Konfigurierbarkeit gegeben.

Zwischen den für die Komponentenorientierung etablierten Technologien Java, CORBA und COM bzw. CLR (siehe dazu Anhänge A.1 mit A.3) ist nur eingeschränkte Interoperabilität möglich. Innerhalb der genannten Technologien ist eine Integration von anderen Systemen gegeben, wenn letztere entsprechende Schnittstellen besitzen.

Komponenten können gleichzeitig als Client und Server dienen, indem sie Dienste zur Verfügung stellen und damit als Server dienen. Die Client-Rolle manifestiert sich beim Bedarf nach weiteren Diensten, die zur Ausführung der selbst angebotenen Dienste benötigt werden. Anpassungen an sich ändernde Rahmenbedingungen sind damit insofern möglich, als zur Laufzeit Implementierungen für die benötigten Dienste identifiziert werden können. Die Spezifikation der benötigten Dienste ist allerdings in der Kontextspezifikation fest vorgegeben. Damit ist Flexibilität bis zu einem gewissen Grad gegeben.

Systemoffenheit kann einer komponentenorientierten Anwendung nur eingeschränkt attribuiert werden, denn die von Komponenten benötigten und bereitgestellten Dienste müssen bereits zum Zeitpunkt der Applikationskonstruktion spezifiziert sein, damit diese anderen Komponenten zur Verfügung gestellt werden können.

Die Kommunikation zwischen den Komponenten erfolgt meist in synchroner Weise, womit auf dieser Ebene eine enge Kopplung gegeben ist. Der Nachrichtenaustausch ist über die Kontextspezifikation an der Schnittstellensemantik orientiert. Die Strukturierung einer Applikation in verteilte Komponenten impliziert keine dezentrale Steuerung der Prozesslogik. Vielmehr werden die Dienste einer Komponente durch ein zentrales Element gesteuert. Werden geeignete Middlewares für die Realisierung der komponentenorientierten Applikation eingesetzt, kann eine dynamische Dienstsuche und -bindung erzielt werden. Auch die Programmiersprachen- sowie Betriebssystemunabhängigkeit ist abhängig von der Wahl der Middleware und beeinflusst damit die Art der Kopplung auf Plattformebene.

Hinsichtlich IS-Architektur lässt sich mit einer komponentenorientierten Architektur zwar durch den Einsatz etablierter Technologien eine Offenheit erzielen. Diese ist aber beschränkt durch die nicht vollständige Interoperabilität zwischen heterogenen Implementierungen. Die Verteiltheit der IS-Architektur ist abhängig von den gegebenen lokalen Restriktionen, aber grundsätzlich über eine komponentenorientierte Architektur realisierbar. Insgesamt ist die gewünschte Flexibilität auf der Ebene der IS-Architekturen nur eingeschränkt erzielbar.

Die Betrachtung von komponentenorientierten Architekturen zeigt, dass Maßnahmen zur Modularisierung und losen Kopplung in eingeschränktem Umfang gegeben sind. Deshalb ist die für die Integration von Informationssystemen gewünschte Flexibilität auf Software-Architekturebene nur bedingt gegeben.

Zusammenfassend lässt sich feststellen, dass die für Integrationsszenarien gewünschte Flexibilität auf den Ebenen der IS- und Software-Architektur über die Komponentenorientierung nur eingeschränkt realisierbar ist.

3.4.2.4 Evaluierung von EAI

IS-Architekturen, die auf der Basis einer EAI gestaltet sind, sind in Abhängigkeit ihrer Implementierung und der dabei gewählten Technologien offen oder integriert. Ziel einer EAI ist zwar, heterogene Systeme zu verbinden, dies wird aber oftmals durch die fehlende Interoperabilität von Technologien beschränkt. Abhängig vom gewählten Integrationsradius (siehe Abschnitt 3.2.4.3) kann eine verteilte oder geschlossene IS-Architektur gestaltet werden. Obwohl technologisch auch weitreichender realisierbar, ist der Integrationsradius einer EAI

meist auf den Bereich innerhalb eines Unternehmens beschränkt. Insgesamt kann festgestellt werden, dass die für Integrationsszenarien gewünschte Flexibilität auf der Ebene der IS-Architektur durch eine EAI durchaus realisierbar ist.

Auf der Ebene der Software-Architektur können hinsichtlich Flexibilität die folgenden Eigenschaften einer EAI identifiziert werden: Eine solche Architektur reduziert die Komplexität von Architekturen zur Integration von Informationssystemlandschaften über die Bereitstellung eines externalisierten Message Brokers. Wird dafür eine geeignete Topologie wie z.B. ein Bus gewählt, ist der Integrationsansatz erweiterbar, um darüber unterschiedliche Technologien vereinen zu können.

Die Interoperabilität ist per se durch den Message Broker gegeben. Werden adäquate Integrationstechnologien eingesetzt, kann die Integration auf den Ebenen Daten, Objekte und Prozessen aus heterogenen Informationssystemen erfolgen.

Die Funktionalitäten, welche von einer EAI-Plattform realisiert werden, können in der Doppelrolle als Diensterbringer und -nutzer implementiert werden. Denn Dienste rufen zur Realisierung ihrer Funktionalität ggf. andere Dienste auf. Mit diesem Mechanismus wird Flexibilität erreicht. Letztere wird durch die Unterstützung unterschiedlicher Integrationstechnologien verstärkt. Außerdem wird beim Einsatz heterogener Technologien die Programmiersprachen- und Betriebssystemunabhängigkeit gewährleistet.

Eine vollständige Systemoffenheit ist in einer EAI nicht gegeben, denn die benötigten Dienste können zwar bei entsprechender Integrationstechnologie erst zur Laufzeit identifiziert und gebunden werden, die benötigten Dienste sind aber ex ante zu spezifizieren.

Der Kommunikationsstil ist abhängig von der gewählten EAI-Plattform. Während z.B. RPC und CORBA jeweils synchrone Aufrufsemantik besitzen, kann die Kommunikation über Web Services asynchron erfolgen. Die verwendeten Integrationstechnologien besitzen meist ein starkes Typsystem. Die Steuerung der Prozesslogik wird durch die aufgerufenen Funktionalitäten des Benutzers bestimmt und erfolgt damit zentral. Werden geeignete Middlewares in einer EAI-Plattform aggregiert, kann eine lose Kopplung auf Plattformebene erreicht werden.

Mit einer EAI wird auf der Ebene der IS-Architektur Flexibilität in Abhängigkeit von der gewählten Implementierung und der lokalen Restriktion erreicht.

Insgesamt lässt sich auf der Ebene der Software-Architektur feststellen, dass durch die Modularisierung Flexibilität erreicht wird. Ausgenommen ist dabei jedoch der Bereich der Systemoffenheit. Durch die Aggregation von unterschiedlichen Integrationstechnologien in der EAI-Plattform wird die Kopplung in wesentlichem Umfang lose gekoppelt. Diese Art der Kopplung wird aber nur auf bestimmten Ebenen erreicht und liegt deshalb bei einer EAI lediglich eingeschränkt vor.

Zusammenfassend lässt sich feststellen, dass mit einer EAI zwar Flexibilität angestrebt wird, diese aber auf der Ebene der IS-Architektur in Abhängigkeit von der Implementierung und auf der Ebene der Software-Architektur nur eingeschränkt gegeben ist.

3.4.2.5 Evaluierung von SOA

Mit einer SOA wird auf der Ebene der IS-Architektur ein offenes System angestrebt. Diese Offenheit wird zwar durch den zentralen Dienstbus unterstützt, ist aber letztendlich abhängig von der gewählten Implementierungstechnologie. Eine SOA zielt weiterhin auf die Gestaltung einer verteilten IS-Architektur ab, um Dienste über die Grenzen von Institutionen anbieten zu können. Mit den beschriebenen Eigenschaften wird auf der Ebene der IS-Architektur Flexibilität erreicht.

In einer SOA werden Dienste über mehrere Schichten der Systemarchitektur und fokussiert auf die zu erbringende Funktionalität in Relation zu einer bestimmten Geschäftsentität zusammengestellt. Über diese Dimension erfolgt die Komposition von Dienstleistungen aus den vorhandenen Diensten. Somit wird von der eigentlichen Implementierung abstrahiert und damit die Komplexität reduziert. Über das Konzept der Dienste kann eine geeignete Erweiterbarkeit und damit eine Modifizierbarkeit der Anwendung erzielt werden.

Die Interoperabilität ist per se in einer SOA gegeben, da in dem Dienstbus eine Vielzahl heterogener Integrationstechnologien vereint wird. Dienste können zur Erfüllung ihrer Aufgaben weitere Dienste aufrufen und bieten selbst Dienste zum Aufruf für andere Komponenten an. Somit kann ein Dienst gleichzeitig als Client und Server dienen.

Die herausragende Eigenschaft einer SOA ist die Flexibilität zur Anpassung an sich ändernde Anforderungen. Dazu können neue Dienste aus bereits bestehenden Diensten flexibel komponiert werden. Dies wird durch das hohe Abstraktionsniveau der Dienstspezifikation erleichtert.

Eine vollständige Systemoffenheit kann einer SOA nicht attribuiert werden: Die von anderen Diensten benötigten Funktionalitäten können zwar bei geeigneter Integrationstechnologie erst zur Laufzeit durch die dynamische Suche und Bindung zur Verfügung gestellt werden, die benötigten Dienste sind aber vorab zu spezifizieren.

In Abhängigkeit von den im Dienstbus gebündelten Technologien erfolgt die Kommunikation in einer SOA synchron oder asynchron. Eine SOA besitzt meist ein starkes Typsystem, weil keine semantisch reichhaltigen Nachrichten über den Dienstbus ausgetauscht werden. Obwohl die Steuerung von Prozessen durch prozessorientierte Dienste verteilt vorliegt, wird erstere dennoch von einer zentralen Instanz angestoßen. Damit kann einer SOA oftmals eine zentrale Steuerung der Prozesslogik attribuiert werden. Weil über den Dienstbus mehrere Integrationstechnologien eingesetzt werden, kann Programmiersprachen- und Betriebssystemunabhängigkeit erreicht werden.

Die Offenheit und Verteiltheit wird bei einer SOA auf Ebene der IS-Architektur im Wesentlichen durch das Dienstkonzept erreicht. Die Ausprägung der Offenheit ist letztendlich abhängig von der gewählten Implementierungstechnologie, wird aber unterstützt durch die Technologie unabhängige Definition von Diensten.

Die Maßnahmen der Modularisierung auf der Ebene der Software-Architektur sind in einer SOA umfassend und erlauben auf allen Ebenen bis auf die Systemoffenheit Flexibilität. Bei der Kopplung lässt sich feststellen, dass diese nur eingeschränkt lose ist.

Insgesamt besitzt eine SOA wegen ihrer Fokussierung auf Dienste ein wesentliches Potenzial bei der Integration von Informationssystemen. Dadurch wird eine weitere Abstraktionsstufe der Implementierungstechnologie eingeführt, um Flexibilität zu gewährleisten.

3.4.2.6 Evaluierung von Web Services

Eine IS-Architektur auf der Basis von Web Services ist offen und verteilt. Damit wird auf dieser Ebene die für Integrationsszenarien gewünschte Flexibilität vollständig unterstützt.

Web Services stellen auf der Ebene der Software-Architektur eine adäquate Basis zur Realisierung einer EAI und damit auch für eine SOA dar. Komplexität wird reduziert, indem relevante Funktionalität über standardisierte Dienstschnittstellen zur Verfügung gestellt wird.

Die Assoziation zwischen Diensterbringer und -nutzer kann dynamisch zur Laufzeit erfolgen. Damit können implementierte Dienste durch alternative Realisierungen ersetzt werden. Weiterhin können mit dem Prinzip der dynamischen Registrierung, Suche und Nutzung von Diensten ggf. Lasten auf andere, aber funktional gleiche Diensterbringer verteilt werden.

Die Interoperabilität ist zwar durch die Existenz von offenen Standards gegeben, jedoch auf den in Abbildung 3.2-7 dargestellten Protokoll-Stack eingeschränkt. Damit ist nur eine begrenzte Interoperabilität gegeben, weil die zu integrierenden Systeme eine Web Services-Schnittstelle implementieren müssen.

Die Aufteilung zwischen Dienstleister und -nutzer ist strikt gehalten. Ein Client verwendet die von einem Provider angebotenen Dienste (siehe auch Abbildung 3.2-8). Damit ist die Architektur in dieser Perspektive inflexibel.

Mit der dynamischen Bindung von Diensten ist zwar eine gewisse Systemoffenheit gewährleistet, diese wird aber durch die Spezifikation von benötigten Diensten vor der Laufzeit eingeschränkt.

Die Kommunikation zwischen Web Services ist abhängig von der gewählten Implementierung synchron oder asynchron. Deshalb ist eine Applikation auf der Basis dieser Technologie hinsichtlich des Kommunikationsstils eng oder lose gekoppelt. Weil die Kommunikation zwischen einem Client und einem Provider auf Nachrichtenaustausch basiert, erfolgt die Kopplung zwischen diesen beiden Komponenten bis zu einem gewissen Grad lose. Obwohl damit eine Nachrichtensemantik vorliegt, kann keine semantische Basis in Form einer gemeinsamen Ontologie identifiziert werden. Deshalb ist die Nachrichtensemantik eingeschränkt. Der Aufruf der Funktionalitäten von Web Services erfolgt zwar dezentral, die Steuerung der Prozesslogik wird aber von der zentralen Instanz ausgelöst. Mit dem in Abbildung 3.2-7 dargestellten Protokoll-Stack für Web Services wird eine Programmiersprachen- und Betriebssystemunabhängigkeit erreicht. Dennoch ist mit der Festlegung auf diesen Stack eine gewisse Einschränkung verbunden, weil die zu integrierenden Systeme diesen zwingend unterstützen oder geeignete Schnittstellen implementieren müssen.

Werden Web Services auf der Basis von offenen Standards realisiert, kann mit ihnen eine offene IS-Architektur gebildet werden. Weiterhin ermöglichen Web Services per se eine verteilte IS-Architektur, weil sie, einen geeigneten Protokoll-Stack vorausgesetzt, auf Internet-Technologien basieren.

Insgesamt kann festgestellt werden, dass auf der Ebene der Software-Architektur die Modularisierung durch Web Services bis zu einem bestimmten Grad gewährleistet wird. Weiterhin ist die Kopplung in einem System aus Web Services eingeschränkt lose, wobei aber die Kommunikation auf dem Austausch von Nachrichten basiert. Die zuletzt genannte Eigenschaft stellt ein wesentliches Differenzierungsmerkmal zu den bisher betrachteten Architekturen dar.

Zusammenfassend kann konstatiert werden, dass auf der Basis dieser Technologie Flexibilität auf den Ebenen der IS- und Software-Architektur erreicht werden kann.

3.4.2.7 Evaluierung von agentenbasierten Systemen

Bei der Konstruktion von IS-Architekturen auf der Basis von Software-Agenten kann ein offenes und per se verteiltes System gestaltet werden. Wesentlich für die Attribuierung der Offenheit ist die Berücksichtigung von Standards wie z.B. FIPA-Protokolle für die Interaktion zwischen Agenten. Die Interoperabilität mit anderen Technologien ist, wie die Ausführungen weiter unten in diesem Abschnitt zeigen, bisher jedoch eingeschränkt.

Beim Einsatz der Agententechnologie wird eine geeignete Agentenplattform vorausgesetzt. Je nach Implementierung werden auf der technischen Ebene unterschiedliche Kommunikationsmechanismen unterstützt, insbesondere auch solche, die die Interaktion über Internet-Technologien erlauben. Insofern realisieren Software-Agenten eine offene und verteilte IS-Architektur, falls Standards und geeignete Technologien eingesetzt werden.

Die Reduktion von Komplexität auf der Ebene der Software-Architektur erfolgt in agentenbasierten Systemen auf der Basis der Aufteilung der Funktionalität in fein granulare Agenten. Diese lassen sich zur Laufzeit zu weiteren Subsystemen zusammenstellen (Jennings 2001, 36-37) und auch wieder dekomponieren. Damit steht bei agentenorientierten Systemen eine flexible und bedarfsorientierte Komposition und Dekomposition von Elementen zu unterschiedlich umfangreichen Subsystemen zur Verfügung. Somit lässt sich in einem agentenbasierten System die Modifizierbarkeit besonderes ausgeprägt realisieren, weil sich diese auf unterschiedlichen Abstraktionsstufen umsetzen lässt.

Die Integration von Altsystemen in ein Agentensystem erfolgt über das Konzept der Wrapper (Jennings 2001, 40). Damit ist die Integration zwar möglich, aber nur mit einem entsprechenden Aufwand zu leisten, denn die Integration in agentenbasierte Systeme wird meist von Informationssystemen nicht unterstützt, weil die Agententechnologie bisher noch keine weite Verbreitung findet. Weiterhin ist die Interoperabilität von Agentensystemen eingeschränkt, weil eine Plattform meist nur einen Standard unterstützt. Außerdem gestaltet sich die Integration einer Agentenplattform in andere Middleware wie CORBA bisher schwierig (Pichler/Plösch/Weinreich 2002, 96-100). Insgesamt ist die Interoperabilität zwar gegeben, aber dennoch eingeschränkt.

Agenten können sowohl als Client als auch als Server fungieren. Ihre Rollenverteilung ist damit durch Gleichberechtigung gekennzeichnet. Zudem können Agenten während ihres Lebenszyklus Rollen annehmen und wieder ablegen sowie zu einem Zeitpunkt in mehreren Rollen agieren. Da Agenten zu Subsystemen komponiert und diese nach Bedarf auch wieder aufgelöst werden können, kann flexibel auf wechselnde Rahmenbedingungen eingegangen werden. Diese Funktionalität kann insbesondere zur Laufzeit erzielt werden. Werden Agentensys-

teme zu einem Peer-to-Peer-Netz erweitert, kann die Lastverteilung dynamisch erfolgen: Sind Komponenten überlastet, können funktional gleichwertige Agenten die Aufgaben übernehmen. Da Agenten außerdem mobil sind, kann die Last auch physikalisch auf mehrere Knoten verteilt werden. Agenten können in diesem Zusammenhang auch das Datentransfervolumen verringern, indem sie zum Speicherort der Daten migrieren und dort ihre Berechnungen ausführen.

Ein Agentensystem ist vollständig systemoffen. Weil sich die für einen Agenten erforderlichen Aktionen beginnend mit der Laufzeit aus den definierten Zielen ergeben, werden erst zu diesem Zeitpunkt relevante Dienste identifiziert. Diese können dynamisch von unterschiedlichen Agenten zur Verfügung gestellt oder auch wieder entfernt werden. Agenten können dynamisch zu einer Agentengruppe oder einem vollständigen Agentensystem hinzugefügt werden. Dies ist wegen der potenziellen Mobilität von Agenten nicht nur auf logischer, sondern insbesondere auch auf physikalischer Ebene möglich.

Die Kopplung in einem Agentensystem ist bis auf die Plattformebene vollständig lose: Die Kommunikation erfolgt in dem Spektrum zwischen synchron (ähnlich Client-Server-Interaktion) und asynchron (soziale Interaktion zur Kooperation, Koordination und Verhandlung über einen Gegenstand). Dabei werden semantisch reichhaltige Nachrichten auf der Basis einer gemeinsamen Ontologie ausgetauscht. Damit ist in einem Agentensystem ein ausgeprägt schwaches Typsystem gegeben. Ein agentenbasiertes System zeichnet sich per se durch einen dezentralen Kontrollfluss aus. Die Steuerung der Prozesslogik ist damit dezentral. Die Dienstsuche und -bindung erfolgen dynamisch zur Laufzeit. Bei der Verwendung von plattformunabhängigen Programmiersprachen kann zumindest vom Betriebssystem abstrahiert werden. Dennoch ist die Plattformunabhängigkeit eingeschränkt, weil Agenten nur innerhalb einer ausgewählten Agentenplattform agieren können.

Werden IS-Architekturen auf der Basis von Software-Agenten konstruiert, können Offenheit bis zu einem gewissen Grad und Verteiltheit vollständig erzielt werden, falls Standards bzw. adäquate Technologien zur Implementierung verwendet werden. Insofern besitzen Software-Agenten auf dieser Ebene ein wesentliches Potenzial zur Unterstützung von Flexibilität in Integrationsszenarien.

Agenten zeichnen sich auf der Ebene der Software-Architektur im Hinblick auf die Maßnahmen zur Modularisierung besonders aus: Sie unterstützen eine zur Laufzeit mögliche und bedarfsorientierte Komposition und Dekomposition von Funktionalitäten aus atomaren oder zusammengesetzten Agentendiensten. Außerdem wird eine vollständige Systemoffenheit erreicht. Auf der Ebene der Plattformunabhängigkeit sind gewisse Einschränkungen gegeben, die durch weitere Arbeiten zur Integration weiterer Middlewares jedoch reduziert werden können.

Bei Agentensystemen ist die Kopplung bis auf die Plattformebene vollständig lose. Einen wesentlichen Anteil daran hat im Vergleich zu anderen Integrationstechnologien die vollständige Nachrichtensemantik, welche durch die Verwendung einer gemeinsamen Ontologie ergänzt wird. Insbesondere durch letztere können Architekturen auf der Basis von Software-Agenten von den übrigen in der vorliegenden Arbeit betrachteten Architekturen differenziert werden.

Somit kann mit einem Agentensystem die für Integrationsszenarien relevante Flexibilität auf den Ebenen der IS- und Software-Architekturen besonders vorteilhaft erreicht werden. Schließlich kann Agenten damit ein wesentliches Potenzial zur Integration von Informationssystemen attribuiert werden.

3.5 Zusammenfassung

In Abschnitt 3 war die Vorteilhaftigkeit von agentenbasierten Systemen im Vergleich zu herkömmlichen Technologien herauszuarbeiten. Dazu wurden für die vorliegende Arbeit relevante Architekturen beschrieben und evaluiert. Ihre Bewertung erfolgte auf den Ebenen der IS- und der Software-Architekturen. Die Erkenntnisse daraus werden im Folgenden beschrieben:

Die für Integrationsszenarien gewünschte Offenheit auf IS-Architekturebene ist meist abhängig von der Auswahl standardisierter Implementierungstechnologien bzw. den Möglichkeiten ihrer Interoperabilität. Werden geeignete Implementierungstechnologien gewählt, ist die Offenheit gegeben. Die Verteiltheit wird maßgeblich durch die lokalen Restriktionen bestimmt. Internet-Technologien erleichtern dabei die Gestaltung einer verteilten IS-Architektur inhärent. Dies ist bei einer angestrebten Informationsintegration über die Grenzen von Institutionen hinaus und insbesondere in wissensintensiven Domänen vorteilhaft. Bspw. erweisen sich in der medizinischen Domäne, in denen das Wissen ständig wächst, solche Informationssysteme als vorteilhaft, die die Informationsintegration unterschiedlicher Datenquellen über das Internet unterstützen, um das Wissen für die Anwender stets aktuell zur Verfügung zu stellen. Die Ergebnisse der Bewertung auf der IS-Architekturebene werden im Folgenden dargestellt:

Um die Komplexität von umfangreichen Software-Systemen beherrschen zu können, werden Software-Architekturen eingesetzt. Im Hinblick auf den Fokus der vorliegenden Arbeit wurden neben grundlegenden Architekturen insbesondere solche für verteilte Systeme bzw. zur Integration von Informationssystemen beleuchtet. Angestrebt wird dabei auch die Herausarbeitung der Vorteilhaftigkeit von agentenbasierten Systemen. Diese Fokussierung ist deshalb gerechtfertigt, weil der Vergleich von MASen und anderen Architekturstilen kaum untersucht ist (Shehory 2000, 89).

Werden integrierte Informationssysteme mit bisherigen Technologien und Architekturen implementiert, können zwei wesentliche Schwachpunkte identifiziert werden: Ungenügende Flexibilität und lediglich deskriptive Beschreibung der einem Informationssystem zugrunde liegenden Informationsmodellierung. Im Folgenden wird aufgezeigt, dass Agentensysteme in der Lage sind, diese Defizite zu beseitigen:

Die Realisierung einer Integrationsarchitektur ausschließlich über z.B. entfernte Prozeduraufrufe kann wesentliche Nachteile mit sich bringen, denn der Einsatz dieser Technologie verlagert die Komplexität der Interoperabilität auf die Entwickler (van Aart et al. 2002): Für eine Implementierung müssen alle Details bezüglich der Signatur, des Verhaltens und der Behandlung von Ausnahmen beachtet werden (van Aart et al. 2002). In diesem Zusammenhang sind auch Web Services in ihren Eigenschaften nicht genügend flexibel (van Aart et al. 2002): Die Kopplung zwischen den zu integrierenden Informationssystemen ist zwar zu einem umfangreichen Anteil lose gestaltet, dennoch ist die auf Nachrichtensemantik basierende Interaktion eingeschränkt, obwohl sich Web Services gerade dadurch von anderen Ansätzen abgrenzen: Ein auf Nachrichtenaustausch basierendes System, wie es in einem MAS gegeben ist, kann als ein nahezu vollständig entkoppeltes, verteiltes System betrachtet werden (van Aart et al.

2002). Die besonders ausgeprägte lose Kopplung in einem solchen System wird wesentlich durch die Nachrichtensemantik und die in Nachrichten ausgetauschten, einer Ontologie folgenden Inhalte ermöglicht. Durch die formale Beschreibung der Nachrichteninhalte können agentenbasierte Systeme wesentlich von anderen Ansätzen differenziert werden. Die Flexibilität ist in einem MAS insbesondere auch zur Laufzeit gegeben, indem bedarfsgerecht dedizierte Subsysteme zur Erfüllung von aktuell relevanten Anforderungen aus atomaren Agenten oder weiteren Subsystemen komponiert und wieder aufgelöst werden. Damit wird eine vollständige Systemoffenheit erreicht.

Lyyntinen/Lehtinen (1984, 38-39) stellen fest, dass herkömmliche Informationssysteme Informationen lediglich deskriptiv beschreiben. Da diese Vorgehensweise als unzureichend betrachtet wird (Lyyntinen/Lehtinen 1984, 38), schlagen *Lyyntinen/Lehtinen* die Modellierung von Informationen vor, die auf der Sprechakttheorie basiert (1984). Auch *Schoop* (1998a) schlägt vor, Informationssysteme auf der Basis der linguistischen und sozialen Theorien (Schoop 1998a) von *Searle* (1969) und *Habermas* (1985) zu gestalten. Dabei wird der intentionale Charakter von Informationen explizit berücksichtigt. Agentensysteme unterstützen diese Art der Informationsmodellierung über die Sprechakt basierte Interaktion inhärent. Deshalb eignen sich Agentensysteme vorteilhaft, die Anforderung einer intentionalen Informationsmodellierung bei der Konstruktion von Informationssystemen geeignet abzubilden.

Insgesamt kann festgestellt werden, dass ein Agentensystem in etablierte Architekturen hineinentwickelt werden kann. Somit können die in der vorliegenden Arbeit dargestellten Architekturen als Metaarchitekturen verstanden werden. Weiterhin wurde aufgezeigt, dass Agentensysteme dabei einen wesentlichen Mehrwert hinsichtlich der durch sie induzierten Flexibilität leisten können. Agentensysteme besitzen außerdem Potenziale auch für zukünftige Entwicklungen im Bereich des ubiquitären oder Organic Computing. Somit kann agentenbasierten Systemen ein hohes Innovationspotenzial zur Beherrschung der zunehmenden Komplexität von Software-Systemen attribuiert werden. Dies geht konform mit der bei der Entwicklung von Informationssystemen erwarteten Orchestrierung von Diensten, die von der technischen Implementierung abstrahieren (Smith/McKeen 2006, 133-134). Agentenbasierte Systeme gehen dabei noch einen Schritt weiter, indem Dienste auch auf autonome Weise erbracht werden können.

4 Betrachtung des Gesundheitswesens

4.1 Überblick

Das Ziel in Abschnitt 4 ist die Ableitung eines Katalogs für Anforderungen an zukünftige Informationssysteme im Gesundheitswesen (Abbildung 4.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1), um Forschungsfrage 1 aus Abschnitt 1.4 zu beantworten.

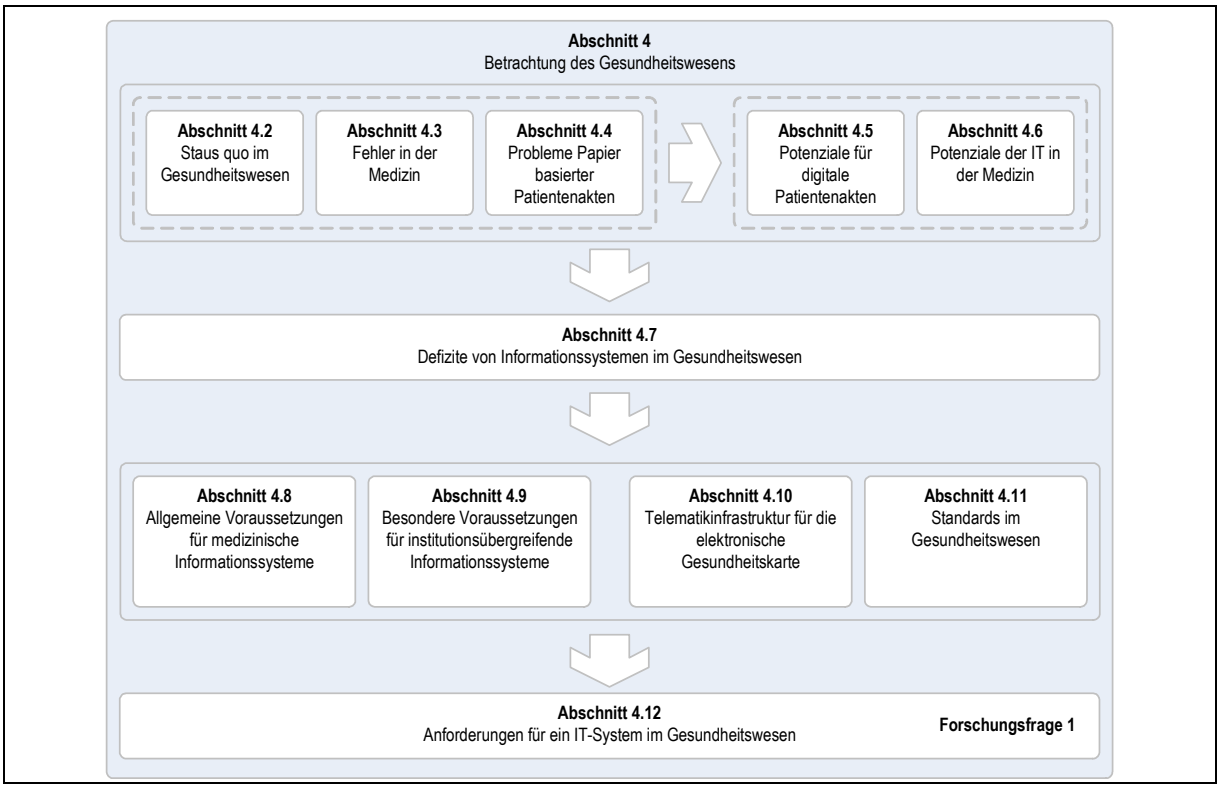

Abbildung 4.1-1: ***Überblick über Abschnitt 4***
Quelle: Eigene Darstellung

Dazu werden zunächst der Status quo im Gesundheitswesen (Abschnitt 4.2), Fehler in der Medizin (Abschnitt 4.3) und Probleme Papier basierter Patientenakten (Abschnitt 4.4) betrachtet. Aus diesen Darstellungen leiten sich die Potenziale für digitale Patientenakten (Abschnitt 4.5) und IT in der Medizin (Abschnitt 4.6) ab. In Abschnitt 4.7 werden die aktuellen Defizite von Informationssystemen im Gesundheitswesen beschrieben, die es verhindern, die identifizierten Potenziale umzusetzen. Aus diesen Defiziten werden die allgemeinen Anforderungen an Informationssysteme im Gesundheitswesen (Abschnitt 4.8) sowie diejenigen für eine Institutionen übergreifende Integration (Abschnitt 4.9) abgeleitet. Weiterhin werden in den Abschnitten 4.10 und 4.11 die für eine Integration erforderliche Kommunikationsinfrastruktur in der Form der TI für die eGK bzw. Standards beschrieben. Diese Anforderungen fließen in den in Abschnitt 4.12 beschriebenen Anforderungskatalog ein.

Grundlage für die Ausführungen in Abschnitt 4 sind die Beschreibungen bei *Lenz et al.* (2005). In der vorliegenden Arbeit werden die dort dargestellten Sachverhalte detailliert ausgeführt und erweitert.

4.2 Status quo im Gesundheitswesen

4.2.1 Überblick über das deutsche Gesundheitswesen

In diesem Abschnitt werden die wesentlichen Akteure und Institutionen im Gesundheitswesen sowie ihre Relationen beschrieben. Dabei wird herausgearbeitet, dass das Gesundheitswesen strukturell bedingt verteilt gestaltet ist.

Das Gesundheitswesen in Deutschland ist im internationalen Vergleich besonders diversifiziert. Zunächst können die Akteure Leistungsempfänger (z.B. Patienten), Leistungserbringer (z.B. Ärzte) sowie Kostenträger (z.B. Krankenkassen) unterschieden werden. Leistungserbringer können weiter in die Bereiche Krankenhäuser (für eine Definition dieser Einrichtung siehe Winter et al. 1998), niedergelassene Ärzte, Apotheken, Pharmakonzerne sowie sonstige Dienstleister unterteilt werden. Letztere untergliedern sich in Heilmittelerbringer wie z.B. Physiotherapeuten, Masseure, Hebammen oder Therapieeinrichtungen. Zu sonstigen Dienstleistern zählen weiterhin Hilfsmittelersteller für z.B. Seh- oder Hörhilfen. Bei niedergelassenen Ärzten ist zwischen Zahnärzten und den übrigen Ärzten zu differenzieren. Krankenhäuser sind üblicherweise in mehrere Bereiche unterteilt, die jeweils (teil-)autonom handeln (Braubach/Pokahr/Lamersdorf 2004, 34). Bei Kostenträgern wird zwischen gesetzlichen und privaten Krankenversicherungen differenziert.

Bei den genannten Leistungserbringern sowie Kostenträgern werden jeweils geeignete Informationssysteme eingesetzt. In Krankenhäusern sind dies z.B. Krankenhausinformationssysteme, welche bei *Winter et al.* (1998) wie folgt definiert werden:

> **„Ein *Krankenhausinformationssystem* (im Original kursiv) ist das Teilsystem eines Krankenhauses, welches alle informationsverarbeitenden Prozesse und die an ihnen beteiligten menschlichen und maschinellen Handlungsträger in ihrer informationsverarbeitenden Rolle umfaßt [!]“.**

Die Relationen zwischen den genannten Akteuren im Gesundheitswesen sind abhängig von der Kategorisierung in private und gesetzliche Kostenträger[14]. Die monetären Flüsse bei privaten Kostenträgern sind in Abbildung 4.2-1 dargestellt und werden im Folgenden erläutert: Nach einem Arztbesuch stellt der Leistungserbringer an den Patienten eine Rechnung über die erbrachten Leistungen. Diese wird vom Patienten beglichen. Um die Kosten vom privaten Kostenträger rückerstattet zu bekommen, reicht der Leistungsempfänger die Rechnung bei seiner privaten Krankenversicherung ein.

[14] Die Grundlagen zu den in diesem Abschnitt beschriebenen Abrechnungsmodalitäten werden von den folgenden Dokumenten gebildet: Beratungsblatt „Wie erhält der Arzt sein Honorar?“ (Stand 01/2006), TK spezial Bayern: Informationen zu Krankenversicherung und Gesundheitswesen (Nr. 2/Juli 2005, S. 11-14). Diese Dokumente wurden freundlicherweise von der Landesvertretung Bayern der Techniker Krankenkasse zur Verfügung gestellt.

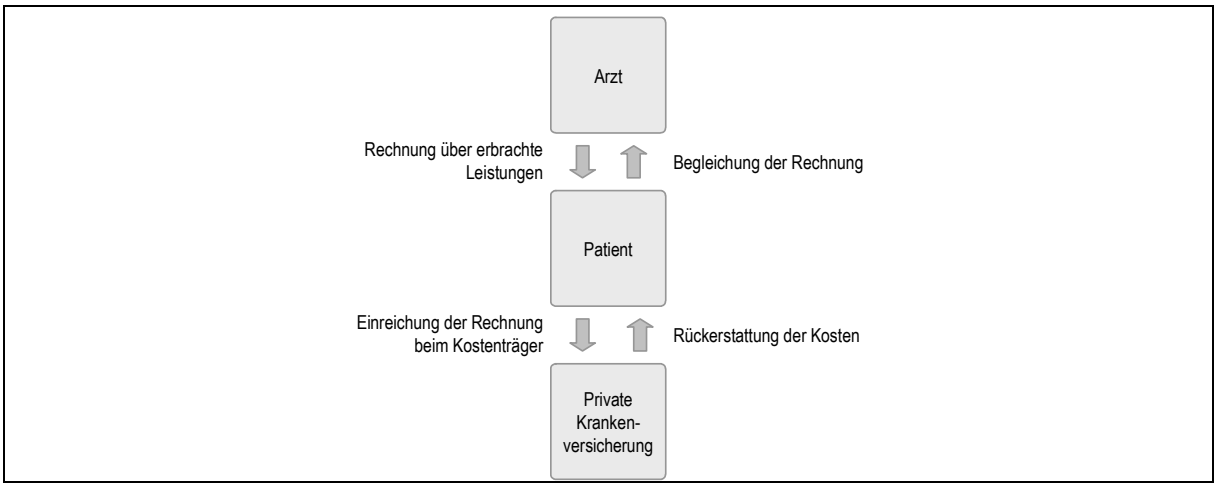

Abbildung 4.2-1: ***Leistungsempfänger, -erbringer und private Kostenträger***
Quelle: Eigene Darstellung

Bei gesetzlichen Krankenversicherungen erfolgt die Kostenübernahme für erbrachte Leistungen in anderer Weise: Bei dieser Konstellation kommen die kassenärztlichen (siehe Abbildung 4.2-2) bzw. kassenzahnärztlichen Vereinigungen hinzu. Damit ist der Patient im Gegensatz zu der Abwicklung bei privaten Kostenträgern in die Abrechnung der Leistungen nicht direkt involviert. Ausgenommen ist hiervon die Leistung der Quartalsgebühr, die vom Patienten an den Leistungserbringer zu entrichten ist. Weiterhin wird der Patient über die Beitragszahlungen an die Krankenkassen in das Abrechnungssystem integriert.

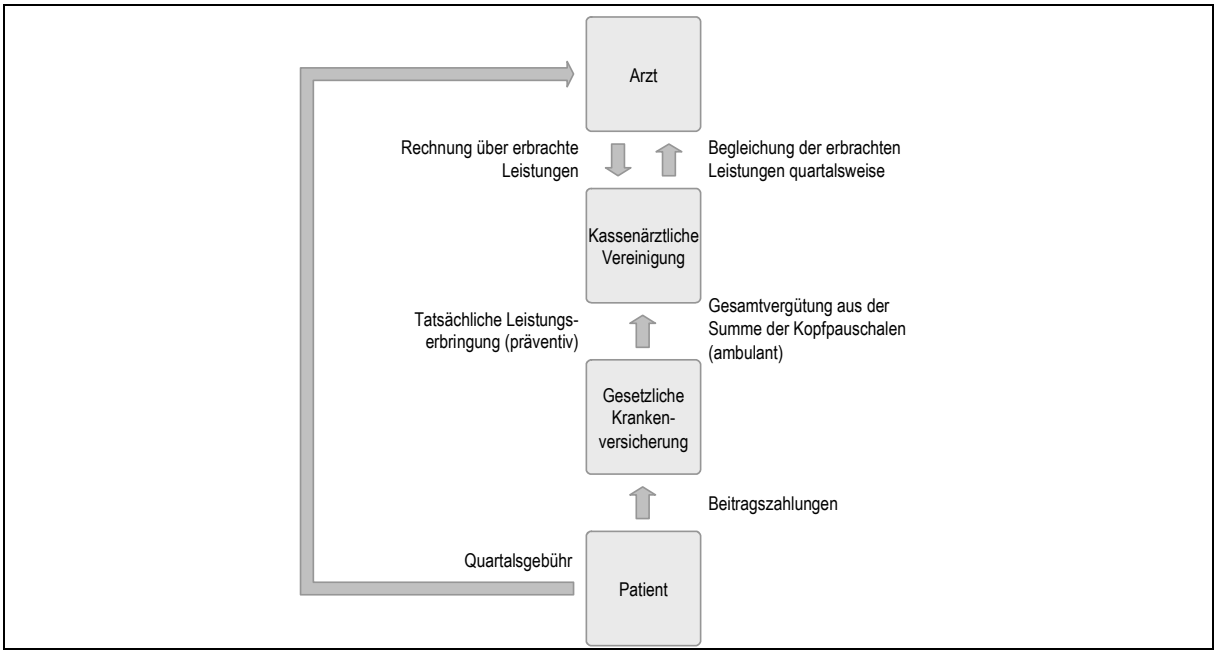

Abbildung 4.2-2: ***Leistungsempfänger, -erbringer und gesetzliche Kostenträger***
Quelle: Eigene Darstellung

Bei präventiven Maßnahmen erfolgt eine feste Vergütung durch die Krankenkassen entsprechend den de facto erbrachten Leistungen. Bei ambulanten Leistungen wird die Abrechnung wie folgt vorgenommen: Durch den Gesetzgeber wird festgelegt, dass die Krankenkasse die

Honorare für Ärzte an die kassenärztliche Vereinigung zahlt. Die Höhe dieses Honorars richtet sich nach den jährlich neu festgelegten Beträgen, die zwischen der Krankenkasse und den kassenärztlichen Vereinigungen vereinbart werden. Pro Versichertem wird dazu ein fester Betrag determiniert. Die Summe dieser Kopfpauschalen kumuliert zur Gesamtvergütung, die von den Krankenkassen an die Vereinigungen entrichtet wird. Grundlage für die Verteilung der Gesamtvergütung sind die von den Ärzten tatsächlich erbrachten Leistungen, die quartalsweise von den Ärzten bei den kassenärztlichen Vereinigungen in Rechnung gestellt werden. Die Verteilung der Gesamtvergütung auf die Ärzte erfolgt über die Bestimmung der Verteilung auf die Facharztgruppen. Diese Verteilung wird von Vertretern der Fachgruppen vorgenommen, die von den Vertragsärzten gewählt werden. Auf diese Verteilung kann von den Krankenkassen Einfluss genommen werden, um mehr Transparenz zu erzielen.

Im Zahnarztbereich stellt sich die Vergütung von Leistungen wie folgt dar: Die Krankenkasse zahlt monatlich eine Abschlagszahlung an die kassenzahnärztliche Vereinigung. Zusätzlich bestimmen die tatsächlichen Einzelleistungen der Zahnärzte die Höhe der monatlichen bzw. quartalsweisen Abschlussrechnungen. Diese Abschlussrechnungen sind nach oben entsprechend dem vorab vereinbarten Budget begrenzt.

Auf der Grundlage des in diesem Abschnitt dargestellten Überblicks über das deutsche Gesundheitswesen werden im folgenden Abschnitt 4.2.2 die besonderen Eigenschaften dieser Domäne beschrieben.

4.2.2 Besondere Eigenschaften der medizinischen Domäne

4.2.2.1 Komplexität und Verteilung

Das Gesundheitswesen zeichnet sich durch die besonderen Eigenschaften der Komplexität und Verteilung aus, die im Folgenden überblicksartig beschrieben werden. Zunächst ist die hohe Komplexität zu nennen (Committee on Quality of Health Care in America: Institute of Medicine 2001, 63-66), welche sich aus der umfangreichen Anzahl von Relationen zwischen den Elementen des Gesundheitswesens ergibt (Committee on Quality of Health Care in America: Institute of Medicine 2001, 63-64). Diese Relationen umfassen die Akteure im Gesundheitswesen, die durch eine verteilte Leistungserbringung gekennzeichnet sind.

Die Komplexität ist aber auch induziert durch die Vielfalt unterschiedlicher Krankheiten (Tan/Wen/Awad 2005, 39). Weiterhin bedingt die demografische Entwicklung eine zunehmende Verschiebung zu einem geriatrischen Schwerpunkt (siehe auch Broy 2005, 7). Die durch die kontinuierlich verbesserte Gesundheitsversorgung steigende Lebenserwartung impliziert eine zunehmende Anzahl von Krankheiten. Diese Komplexität im Gesundheitswesen erfordert eine Spezialisierung der Aufgabenträger auf ausgewählte Bereiche (Simoneit 1998, 55).

Der hohe Grad an Spezialisierung bedingt wiederum die beobachtbare Arbeitsteilung im Gesundheitswesen und damit auch die örtliche Diversifizierung der Leistungserbringung. Damit sind üblicherweise an einem Behandlungsprozess mehrere Akteure beteiligt. Die Beteiligung unterschiedlicher Akteure kann dazu führen, dass wertvolle Zeit oftmals für die Suche nach Dokumenten oder auch medizinischen Geräten verbraucht wird (Volpp/Grande 2003, 853).

Die Verteilung des Behandlungsprozesses sowie die umfangreiche Anzahl der Beteiligten und Teilprozesse (Leape 1997, 221) bedingen die Notwendigkeit von geeigneten Kommunikations-, Kooperations- und Koordinationsbeziehungen zwischen den am Behandlungsprozess beteiligten Akteuren, die insbesondere auch auf Krankenhäuser zutrifft (Lenz et al. 2002, 571). Dennoch kann beobachtet werden, dass die jeweiligen Leistungsträger in der Regel autonom handeln. Wegen des dezentralen Charakters der Organisationsstruktur z.B. in einem Krankenhaus ist oftmals keine zentrale Koordination eines gesamten Behandlungsprozesses (Leape 1997, 221) möglich. Dies impliziert wiederum Brüche in dem gewünschten durchgängigen Prozess, der eine enge Kooperation der verteilten Leistungsträger erfordert.

Mit dieser Darstellung können die Eigenschaften der Komplexität und Verteilung sowie die daraus induzierten Konsequenzen wie folgt zusammengefasst werden:

- *Hohe Komplexität induziert durch umfangreiche Anzahl der Akteure und Krankheitsbilder*
- *Spezialisierung der Leistungserbringer und damit Diversifizierung der Leistungen*
- *Verteilung bedingt geeignete Kommunikation, Kooperation und Koordination*
- *Vielzahl und unzureichende IT-Unterstützung (Avison/Young 2007, 73) der Interaktionen erschwert die Effektivität und Effizienz der Leistungserbringung*

Neben diesen eher statischen Eigenschaften kommt im Gesundheitswesen eine ausgeprägte Dynamik hinzu, welche den Leistungserbringungsprozess zusätzlich erschwert. Diese wird im folgenden Abschnitt 4.2.2.2 beschrieben.

4.2.2.2 Dynamik

Die für den eigentlichen Behandlungsprozess hinsichtlich der Genesung relevanten Akteure im Gesundheitswesen, Patienten und Leistungserbringer, sind veränderlich, deshalb ist das gesamte System dynamisch (Committee on Quality of Health Care in America: Institute of Medicine 2001, 64). Die Vielzahl vorhandener Krankheiten, ihre mannigfaltigen Behandlungsmaßnahmen sowie die unterschiedlichen Reaktionen der Patienten auf diese Maßnahmen (Tan/Wen/Awad 2005, 39) induzieren eine Unvorhersagbarkeit hinsichtlich der erforderlichen Prozesse. Diese nicht eindeutige Antizipierung von Handlungen innerhalb des Systems bedingt eine Variabilität im Leistungserbringungsprozess (Committee on Quality of Health Care in America: Institute of Medicine 2001, 64). Hinzu kommen sich dynamisch ändernde Rahmenbedingungen in Versorgungsprozessen. Veränderungen treten z.B. durch unvorhersagbare Ereignisse wie Notfälle auf.

Die im Gesundheitswesen beobachtbare Dynamik von Prozessen (Kirn et al. 2000, 1) erschwert die Leistungserbringung wesentlich, weil bei der Bereitstellung der erforderlichen Informationen zum Behandlungszeitpunkt mit nicht antizipierbaren Ereignissen zu rechnen ist. Bereits die medienneutrale Betrachtung von Informationssystemen im folgenden Abschnitt 4.2.2.3 zeigt, dass durch bisher eingesetzte Informationssysteme relevante Informationen nicht zur Verfügung gestellt werden können, obwohl bei dieser Darstellung die Dynamik ausgeblendet wird. Damit wird deutlich, dass grundlegende Veränderungen zur geeigneten Bereitstellung von Informationen erforderlich sind.

4.2.2.3 Medienneutrale Betrachtung von Informationssystemen

Im Folgenden werden wesentliche Eigenschaften von Informationssystemen im Gesundheitswesen sowie ihre Auswirkungen auf die Behandlungsqualität beleuchtet: Diese Systeme sind in der Regel verteilt und heterogen (McDonald 1997, 213). Diese Tatsache wird einerseits durch die in Abschnitt 4.2.2.1 beschriebene lokale Verteilung der Leistungserbringer verur-

sacht. Andererseits bedingt auch die Spezialisierung der Leistungserbringer die Heterogenität der Informationssysteme. So setzen z.B. einzelne Abteilungen in einem Krankenhaus Spezialsysteme ein, die für ihre jeweiligen Verwendungszwecke am besten geeignet erscheinen (McDonald 1997, 216).

Diese Verteilung der Informationssysteme ist sowohl innerhalb einer Institution als auch insbesondere über Institutionsgrenzen hinweg beobachtbar. Die Verteilung von Informationen erschwert die Bereitstellung relevanter Daten. Deshalb werden die Daten der eingesetzten Spezialsysteme oftmals integriert, wozu geeignete Architekturen eingesetzt werden (siehe Abschnitt 4.6.1). Für den dadurch angestrebten Datenaustausch von medizinischen und administrativen Informationen sind geeignete syntaktische und semantische Standards erforderlich (siehe dazu Abschnitt 4.11).

Die strikte Trennung zwischen ambulantem und stationärem Sektor, aber auch die Auswahl heterogener Systeme innerhalb einer Institution haben zur Folge, dass zur Informationsdarstellung und Speicherung nicht nur verteilte Systeme, sondern oftmals auch unterschiedliche Medien eingesetzt werden. Solche Medienbrüche können durch die folgenden Tatsachen verursacht werden:

- Paralleler Einsatz von Papier basierten Dokumenten und digitalen Informationssystemen
- Unterschiedliche Arten von Papierdokumenten
- (Fern-) mündliche und Papier basierte Kommunikation
- Einsatz von heterogenen Informationssystemen

Als Folge von Medienbrüchen werden durch die Datenübertragung in andere Medien zusätzlicher Arbeitsaufwand und Inkonsistenzen in den erfassten Daten beobachtet (Mikkelsen/Aasly 2001). Obwohl bei der von *Mikkelsen/Aasly* durchgeführten Studie (2001) teilweise signifikante Differenzen zwischen den in den jeweiligen Medien vorgehaltenen Informationen identifiziert werden, wird nur ein geringer Anteil an fehlerhaften Dokumenten im Vergleich zur Gesamtzahl der untersuchten Dokumente festgestellt. Die potenziellen Konsequenzen einer falschen Information werden dabei jedoch schwerwiegender eingeschätzt als vollständig fehlende Information. Deshalb sind geeignete Maßnahmen zu ergreifen, um eine durchgängige und vollständige Informationsbereitstellung zu ermöglichen.

Aus dieser Darstellung wird ersichtlich, dass das deutsche Gesundheitswesen in seiner Struktur verteilt gestaltet ist sowie durch Komplexität und Dynamik gekennzeichnet ist. Diese Eigenschaften erschweren die Bereitstellung von Informationen, die für einen Behandlungsprozess relevant sind. Bspw. entstehen aus der Verteilung Medienbrüche, die wiederum Übertragungsfehler induzieren und damit zu fehlender Information im Versorgungsprozess führen können. Damit wird insgesamt die Behandlungsqualität reduziert. Es sind deshalb geeignete Konzepte erforderlich, die die genannten Eigenschaften der Domäne Gesundheitswesen beherrschen und schließlich zu Effektivitäts- und Effizienzsteigerungen führen können.

Ein Ansatz, um die sich aus der Verteiltheit ergebenden Problemstellungen zu bewältigen, findet sich in medizinischen Versorgungszentren (Vera 2006). In dieser Sonderform werden mehrere Leistungserbringer wie niedergelassene Ärzte, Krankenhäuser und weitere Gesundheitsdienstleister zusammengeschlossen, die gemeinsam Patienten behandeln. Angestrebt wird dabei eine durchgängige, am Patienten ausgerichtete Versorgung. Bei dieser Form der integrierten Versorgung nimmt die Patientenakte als wesentliches Kommunikations- und Informations-

medium einen zentralen Status ein. Um die genannten Problemstellungen von Patientenakten zu beseitigen, sind die darin beschriebenen Informationen auf einem geeigneten Medium zur Verfügung zu stellen. Um einen solchen Informationsträger identifizieren zu können, werden als Vorarbeit im folgenden Abschnitt 4.2.3 zunächst die Syntax und Semantik von Patientenakten beleuchtet.

4.2.3 Syntax und Semantik einer Patientenakte

4.2.3.1 Syntax

Patientenakten stellen im Leistungserbringungsprozess ein wesentliches Medium dar. Dabei erfüllen erstere die Aufgaben der Dokumentation der erbrachten Leistungen, aber auch die Abstimmung zwischen pflegerischem und medizinischem Personal. Somit besitzt eine Patientenakte neben medizinischen auch koordinative Daten. Diese Daten können, unabhängig vom Medium in unterschiedlichen Formaten dargestellt werden, die im Folgenden beschrieben werden:

Wyatt/Wright (1998, 1376) unterscheiden wesentliche Formate, die in Tabelle 4.2-1 jeweils mit ihren Vor- und Nachteilen dargestellt sind. Weil die Eignung der Darstellungsformen für den jeweiligen Zweck vom Format abhängig ist, ist für die Wahl des Formats der Verwendungszweck zu berücksichtigen, weil damit im jeweils vorliegenden Kontext die Suche und Interpretation der passenden Information vereinfacht werden können (Wyatt/Wright 1998, 1376).

Die prosaische Beschreibung von Inhalten und Bildmaterial werden am meisten verwendet (Baud et al. 1998; Wyatt/Wright 1998, 1376). Obwohl die Daten in diesem textuellen Format einfach erfasst werden können, sind die benötigten Informationen schwierig zu identifizieren. Bereits die Darstellung in einer strukturierten Form, bspw. in einer Tabelle, kann das Auffinden der relevanten Daten verbessern. In einem Experiment (Fries 1974) konnte gezeigt werden, dass die Darstellung von Informationen in Tabellen die Identifikation der benötigten Daten tatsächlich beschleunigt oder überhaupt erst ermöglicht.

Datenformat	Vorteile	Nachteile
Handschriftliche Prosa	• Einfach und schnell zu schreiben • Flexibel, sinnbehaftet	• Schwierig zu lesen und zu durchsuchen • Umfangreich, unstrukturiert, mehrdeutig
Getippte Prosa	• Einfach und schnell zu diktieren • Flexibel	• Schwierig zu transkribieren und zu durchsuchen • Umfangreich, unstrukturiert
Strukturierter Text	Einfacher zu durchsuchen	• Schwieriger zu schreiben oder zu diktieren • Weniger flexibel
Tabellen	• Kompakt • Einfach zu durchsuchen zum Anstellen von Vergleichen	• Schwierig zu erstellen • Unflexibel
Diagramme	• Verdeutlichung von zwei- oder dreidimensionalen und anderen Relationen • Nicht mehrdeutig	• Eine geeignete Darstellung gestaltet sich schwierig. • Beanspruchung von Platz
Grafiken	Verdeutlichung von Trends	• Eine geeignete Darstellung sowie ein adäquater Entwurf gestalten sich schwierig. • Beanspruchung von Platz

Animationen, interaktiv oder Bewegtbilder	Verdeutlichung von Trends und Zusammenhängen	• Setzen IT-Unterstützung voraus • Teuer zu entwickeln

Tabelle 4.2-1: ***Vor- und Nachteile von Datenformaten in Patientenakten***
Quelle: In Anlehnung an *Wyatt/Wright* (1998, 1376)

Diese Ausführungen zeigen, dass zur Steigerung der Effizienz und Effektivität der Informationssuche Daten in einer Patientenakte in einem geeigneten Format darzustellen sind. Bereits die Strukturierung erleichtert die Identifikation der jeweils benötigten Informationen. Nicht nur eine fehlende Strukturierung von Daten, sondern auch unterschiedliche semantische Belegungen können die Datenerfassung und -weiterleitung erschweren. Dies wird im folgenden Abschnitt 4.2.3.2 dargestellt.

4.2.3.2 Semantik

Die Interpretation von Daten wird maßgeblich bestimmt durch die Zuordnung von Semantik zu den Termini. Dafür ist eine einheitliche Abbildung Voraussetzung. Diese ist jedoch nicht immer gegeben, obwohl sie für einen Informationsaustausch zwischen Leistungserbringern erforderlich ist. Im Folgenden werden Ursachen unterschiedlicher semantischer Belegungen und mögliche Lösungsansätze beschrieben.

Das Entstehen unterschiedlicher Terminologien kann bereits durch unterschiedliche Curricula bedingt sein. Denn die „Verwendung von beschreibenden Begriffen beruht auf Gewohnheiten, auf Vermittlung durch bestimmte klinische Schulen, die auch in entsprechenden Standardwerken zum Ausdruck kommen, auf regional und überregionalen [!] akzeptierten Klassifikationen […] und auf bereits bestehenden Konsensusergebnissen“ (Heldwein et al. 1999, 3).

Die Möglichkeit der unterschiedlichen Interpretation von Informationen ist nicht nur bei textuellen Informationen gegeben, sondern auch bei der Betrachtung anderer Medien. Die Notenwendigkeit zur Standardisierung entsteht somit auch bei der Evaluierung der Ergebnisse von Bild gebenden Verfahren. Ein verbaler Terminologiestandard „garantiert jedoch nicht, daß [!] wirklich alle Anwender mit dem jeweiligen Begriff dieselben visuellen Vorstellungen verbinden“ (Heldwein et al. 1999, 6). Als Lösung wird von *Heldwein et al.* (1999, 6) vorgeschlagen, geeignete Bilder bei den jeweiligen Termini zu hinterlegen, um die korrekte Assoziation von Begriffen und Bildern zu unterstützen. Somit wird gewährleistet, dass Synonyme und Homonyme zu einer einheitlichen Interpretation der dargestellten Sachverhalte und nicht wegen unterschiedlicher semantischer Belegung zu Missverständnissen führen.

Eine einheitliche Interpretation der eingesetzten Medien ist nicht nur bei der Kommunikation der Leistungserbringer erforderlich. Die automatisierte Weiterverarbeitung von Informationen setzt vielmehr eine einheitliche Interpretation der Sachverhalte und damit eine Erschließung der Semantik (vlg. dazu Abschnitt 4.11.3) zwingend voraus. So ist die Festlegung auf eine durchgängige Terminologie eine wesentliche Grundvoraussetzung für den Einsatz von z.B. Entscheidungsunterstützungssystemen (Lenz et al. 2005, 110).

Die Gestaltung einer einheitlichen Terminologie als Grundlage für die IT-gestützte Weiterverarbeitung ist jedoch eine besondere Herausforderung, denn von „Auswahl und Umfang der Befundbegriffe hängt ab, ob die automatischen Texte ausreichend differenziert, variabel und

genau sind und damit in der Mehrzahl der Fälle nicht nachträglich ergänzt oder geändert werden müssen“ (Heldwein et al. 1999, 4).

Bereits der Austausch von Daten zwischen Leistungserbringern erfordert ein einheitliches semantisches Verständnis. Die Notwendigkeit der Vereinheitlichung wird bei einer angestrebten automatisierten Weiterverarbeitung noch deutlicher. Deshalb ist eine Erarbeitung von geeigneten semantischen und auch syntaktischen Standards wesentlich, um eine integrierte Versorgung mit IT-Unterstützung zu ermöglichen. Ein Überblick über relevante Standards und ihre mögliche Integration wird in Abschnitt 4.11 dargestellt.

Die bis dato beobachtbare Diversifizierung im Gesundheitswesen verursacht eine zu wenig ausgeprägte Durchgängigkeit von Daten, Prozessen und geeigneter IT. Insbesondere kann bei der Dokumentation eine Vereinheitlichung auf syntaktischer und semantischer Ebene nicht festgestellt werden. Dabei ist auch eine ungenügende Ausrichtung am Patienten beobachtbar. Eine wesentliche Voraussetzung zur Beseitigung der genannten Problemstellungen ist in der Betrachtung von Prozessen aus einer anderen Perspektive gegeben, welche im folgenden Abschnitt 4.2.4 vorgenommen wird.

4.2.4 Prozesssicht im Gesundheitswesen

Im Gesundheitswesen werden Prozesse wegen der inhärenten Verteilung bislang z.B. an Abteilungen oder Institutionen ausgerichtet. Damit wird eine durchgängige Behandlung des Patienten erschwert. Eine ganzheitliche, am Patienten ausgerichtete Sicht kann in der aktuellen Situation nur in Einzelfällen (z.B. in Versorgungszentren, siehe dazu Abschnitt 4.2.2.3) beobachtet werden. Im Folgenden werden, ausgehend von einer Übertragung von Produktionsprozessen in das Gesundheitswesen, für eine Prozessreorganisation erforderliche Veränderungen sowie mögliche sich daraus ergebende Potenziale beleuchtet.

In Anlehnung an Produktionsprozesse in der Industrie können Prozesse im Gesundheitswesen wie folgt interpretiert werden (Elson/Faughnan/Connelly 1997, 268-269): Der Produktionsprozess entspricht dem Entscheidungsfindungsprozess im Gesundheitswesen (siehe dazu auch Abbildung 4.2-3), wobei die produzierten Güter Entscheidungen über die weitere Behandlung des Patienten repräsentieren. Eine wichtige Grundlage für diese Entscheidungen bilden das medizinische Fachwissen (Knowledge) und die Daten über die bisherige Patientenhistorie (Patient Data, Huth 1985; Smith 1996; Elson/Faughnan/Connelly 1997, 266). Das entscheidungsrelevante Wissen wird sowohl dem Gedächtnis des Leistungserbringers (Long Term Memory) als auch externen Quellen (External Memory) entnommen. Dem Leistungserbringer im Produktionsprozess entspricht im Gesundheitswesen der Träger der relevanten Entscheidungen (Elson/Faughnan/Connelly 1997, 266).

Aus dieser Prozesssicht kann die Entscheidungsfindung eines Leistungserbringers wie folgt beschrieben werden: Sucht ein Patient einen Gesundheitsdienstleister auf, werden zunächst relevante Informationen wie aktuelle Beschwerden oder Symptome (Inputs, Perception, Attention) in Erfahrung gebracht. Diese Informationen werden mit dem eigenen Wissen des Leistungserbringers (Long Term Memory) und weiteren Informationen aus externen Speicherorten wie Literatur oder der Patientenakte (External Memory) verknüpft. Abschließend werden aus den verknüpften Informationen Schlussfolgerungen gezogen, die zur eigentlichen Entscheidung führen. Dabei wird ersichtlich, dass Informationen einen wesentlichen Einfluss auf die korrekte Entscheidung (Elson/Faughnan/Connelly 1997, 268) und damit auf die Qualität des Behandlungsprozesses besitzen. Deshalb sind diese Informationen nach dem informa-

tionslogistischen Prinzip (siehe dazu Abschnitt 4.2.6) zur Verfügung zu stellen (Leape 1997, 221), um die richtigen Voraussetzungen für den Entscheidungsprozess zu schaffen.

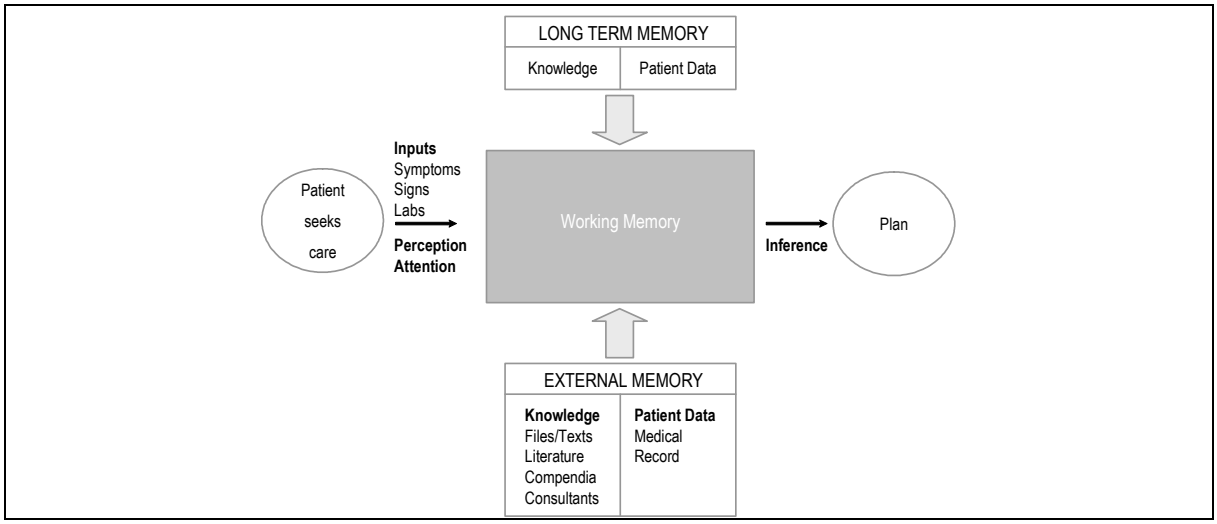

Abbildung 4.2-3: ***Prozesssicht für eine Entscheidung eines Leistungserbringers***
Quelle: In Anlehnung an *Elson/Faughnan/Connelly* (1997, 269)

Die beschriebene Prozesssicht auf die Interaktion eines Leistungserbringers mit seinem Patienten fokussiert nur einen Teil des gesamten Behandlungsprozesses. Ziel der angestrebten Prozesssicht im Gesundheitswesen ist, die abteilungs- oder institutionsinterne Sicht (von Eiff/Ziegenbein 2003, 57) zugunsten einer Patientenorientierung (Berger 2004, 45) aufzugeben. Im Zentrum steht dabei der Patient, der im Laufe einer ganzheitlichen Behandlung unterschiedliche Abteilungen durchläuft. In einem weiteren Schritt wird die gesamte Behandlung auch über Institutionsgrenzen hinweg als durchgängiger Prozess betrachtet (Beyer et al. 2004, 264 und 269), in dessen Zentrum der Patient mit der jeweils erforderlichen Behandlung steht.

Die dafür nötige Prozessreorganisation bietet das Potenzial, bisherige Abläufe zu überdenken und ggf. neu zu strukturieren, um sich den Änderungen im Gesundheitswesen anzupassen sowie Effizienz- und Effektivitätssteigerungen zu erzielen. Diese Möglichkeit wird insbesondere mit der Einführung der elektronischen Gesundheitskarte (eGK) gegeben. Dabei kann ein wesentliches Potenzial identifiziert werden, wenn aus den sich ändernden Abläufen auch Vorteile für die jeweiligen Institutionen generiert werden können, indem die Möglichkeiten der eGK, die insbesondere durch die freiwilligen Anwendungen gegeben sind (Bernnat/Booz Allen Hamilton GmbH 2006, 24, 27 und 195-264), vollständig ausgeschöpft werden. Dazu sind die Informationssystemlandschaft und ihre Architektur geeignet anzupassen, um den Anforderungen des informationslogistischen Prinzips (siehe Abschnitt 4.2.6) zu genügen.

Die Darstellungen in diesem Abschnitt zeigen, dass eine Ausrichtung der Prozesse am Patienten für die Gewährleistung eines durchgängigen Behandlungsprozesses erforderlich ist. Dazu sind bisherige Prozesse sowie die eingesetzten Informationssysteme so anzupassen bzw. derart zu gestalten, dass eine Informationsbereitstellung entsprechend dem informationslogistischen Prinzip erfolgen kann. Die in der Medizin zur Verfügung zu stellenden Informationen besitzen jedoch besondere Eigenschaften, die die Realisierung dieses Prinzips erschweren. Diese Charakteristika von Information werden im folgenden Abschnitt 4.2.5 beschrieben.

4.2.5 Eigenschaften der Information in der Medizin

Wie bereits in Abschnitt 4.2.4 beschrieben, besitzt Information einen wesentlichen Stellenwert beim Prozess der Entscheidungsfindung zur weiteren Behandlung des Patienten: Entscheidungen in der Medizin werden auf der Basis der zum Zeitpunkt der Behandlung bzw. der im Langzeitgedächtnis des Leistungserbringers verfügbaren Informationen getroffen (siehe auch Elson/Faughnan/Connelly 1997, 268-269). Diese Entscheidungsfindung wird jedoch durch die besonderen Eigenschaften der Information in der Medizin erschwert, welche im Folgenden beschrieben werden:

Der Umfang verfügbarer Information, insbesondere in digitaler Form, nimmt stark zu (Lehmann/Spitzer 2006, 3). Die Informationsmenge wächst dabei in zwei Dimensionen (Simoneit 1998, 92): Durch neue Forschungserkenntnisse nehmen die für die Entscheidungsfindung relevante medizinische Wissensbasis und damit die zu sichtende wissenschaftliche Publikationsmenge kontinuierlich zu (Simoneit 1998, 92). Dieses Wissen besitzt eine hohe Relevanz für den klinischen Alltag, denn es werden bspw. durch die evidenzbasierte Medizin Leitfäden für eine standardisierte Behandlung erarbeitet (Spreckelsen et al. 2006, 34). Um dieser sich ständig weiterentwickelnden medizinischen Wissensbasis gerecht zu werden, ist die Beschäftigung mit neuen Forschungsergebnissen erforderlich. Es konnte gezeigt werden, dass das ausführliche Studium von adäquater Literatur dazu beitragen kann, dass hinsichtlich Qualität und Kosten effizientere Entscheidungen getroffen wurden (Cambliss/Conley 1996). Dabei ist jedoch vorauszusetzen, dass die relevante Literatur bzw. die in ihr enthaltene Information in geeigneter Form zur Verfügung steht. Die Suche nach adäquaten Publikationen kann sich jedoch umfangreich gestalten (Ely et al. 2002). Auch neue diagnostische und therapeutische Maßnahmen vergrößern die zu bewältigende Informationsmenge (Simoneit 1998, 70), insbesondere Bild gebende Verfahren generieren umfangreiches Datenmaterial.

Die weiter oben beschriebenen Eigenschaften des Gesundheitswesens, die steigende Arbeitsteilung, die damit verbundene Anzahl der Leistungserbringer, ihre Verteiltheit und neue Untersuchungsmethoden tragen selbst dazu bei, dass Informationen auch hinsichtlich einer zweiten Dimension zunehmen, die die Umfänge der Krankenakten wachsen lassen (Simoneit 1998, 93): Die zunehmende Spezialisierung der Leistungserbringer bedingt die daraus resultierende Arbeitsteilung und damit die Notwendigkeit zur Koordination, Kooperation und Kommunikation des Behandlungsteams. Dies impliziert wiederum die Zunahme der auszutauschenden Informationen, um den anderen am Behandlungsprozess Beteiligten die erforderlichen Informationen zukommen zu lassen.

Die beschriebene Informationsflut erschwert die manuelle Sichtung der verfügbaren Informationen. Dies kann auch zum Auftreten von medizinischen Fehlern führen, wenn in der zur Verfügung stehenden Zeit eine fundierte Entscheidung zu treffen ist (siehe dazu auch Abschnitt 4.3). Es reicht also nicht aus, Informationen nur zur Verfügung zu stellen, sondern es müssen die Daten entsprechend dem informationslogistischen Prinzip (siehe Abschnitt 4.2.6) selektiert, interpretiert und miteinander verknüpft werden, um aus der vorhandenen Information neue Erkenntnisse zu gewinnen (Lehmann/Spitzer 2006, 3) und eine geeignete Entscheidung zur weiteren Behandlung des Patienten zu treffen. Bisher verfügbare Informationssysteme im Gesundheitswesen unterstützen dieses Prinzip jedoch meist nur unzureichend. Deshalb sind geeignete Informationssysteme auf der Grundlage einer adäquaten Implementierungstechnologie zu entwickeln, die Informationen nach diesem Prinzip zur Verfügung stellen, welches im folgenden Abschnitt 4.2.6 beschrieben wird.

4.2.6 Informationslogistik

Der Begriff der Informationslogistik entwickelte sich aus den fünf Gesetzen der Bibliothekswissenschaften (Ranganathan 1931). *Ranganathan* (1931) beschreibt diese wie folgt:

- „BOOKS ARE FOR USE [im Original in Großbuchstaben]“ (Ranganathan 1931, 1, die vollständige Darstellung des ersten Gesetzes findet sich auf den S. 1-73).
- „EVERY PERSON HIS OR HER BOOK [im Original in Großbuchstaben]“ (Ranganathan 1931, 75 und 74-298)
- „EVERY BOOK ITS READER [im Original in Großbuchstaben]“ (Ranganathan 1931, 299 und 299-335)
- „SAVE THE TIME OF THE READER [im Original in Großbuchstaben]“ (Ranganathan 1931, 337 und 336-381)
- „A LIBRARY IS A GROWING ORGANISM“ (Ranganathan 1931, 383 und 383-416)

Diese Prinzipien beschreiben diejenigen Eigenschaften einer Bibliothek, die die Leser dabei unterstützen sollen, die gewünschten Informationen zu erhalten bzw. zu finden. Das informationslogistische Prinzip (Abbildung 4.2-4) kann damit als eine Verallgemeinerung der Verfügbarkeit von Informationen in Büchern eines Bibliotheksbestandes auf eine medienunabhängige Verfügbarkeit von Informationen betrachtet werden.

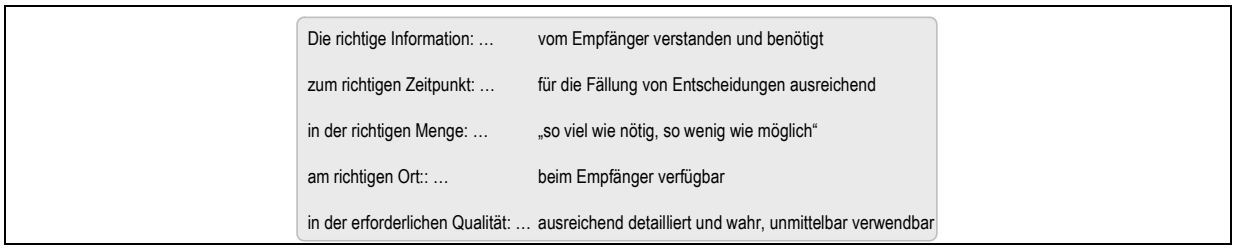

Abbildung 4.2-4: ***Informationslogistisches Prinzip***
Quelle: *Augustin* (1990, 23), geprägt durch die fünf Gesetze der Bibliothekswissenschaften (Ranganathan 1931)

Übertragen auf die medizinische Domäne impliziert das informationslogistische Prinzip, dass Informationen insbesondere auch hinsichtlich der in Abschnitt 4.2.5 beschriebenen, kontinuierlich steigenden und zu bewältigenden Informationsflut für den Entscheidungsträger im Behandlungsprozess zur Verfügung zu stellen sind, damit Informationen als geeignete Basis der in Abschnitt 4.2.4 dargestellten Entscheidungsfindung fungieren können. Ziel ist es dabei, die Informationen bedarfsorientiert bereitzustellen. Sind die relevanten Informationen verfügbar, kann die Qualität der Entscheidungsfindung und damit auch letztendlich des Behandlungsprozesses erhöht werden.

Die Umsetzung des informationslogistischen Prinzips ist zusammenfassend eine wesentliche Voraussetzung für die Verbesserung der Effektivität, Effizienz und Qualität von Behandlungsprozessen. Sie erhält insbesondere dann zusätzliche Bedeutung, aber auch einen umfangreicheren Komplexitätsgrad, wenn die Informationen über Institutionsgrenzen hinweg verfügbar gemacht werden sollen. Das informationslogistische Prinzip ist damit ein wesentlicher Faktor für die angestrebte durchgängige, integrierte und am Patienten ausgerichtete Versorgung. Letztere wird im folgenden Abschnitt 4.2.7 beschrieben. Dabei ist auch die Ebene der Informationssysteme rele-

vant, denn geeignete Systeme bilden die Voraussetzung zur Realisierung des informationslogistischen Prinzips über Institutionsgrenzen hinweg.

4.2.7 Durchgängige, integrierte Versorgung

Ausgehend von der Definition des Begriffes integrierte Versorgung werden im Folgenden Voraussetzungen und erwartete Nutzenpotenziale beschrieben:

Die integrierte Versorgung setzt die Realisierung des informationslogistischen Prinzips voraus. Der Begriff der integrierten Versorgung (Schweiger et al. 2007a) wird deshalb wie folgt definiert:

> Die integrierte Versorgung wird verstanden als vertikal und horizontal durchgängige IT, Prozesse und Daten. Diese Durchgängigkeit wird nicht nur innerhalb von Institutionen, sondern insbesondere über deren Grenzen hinweg angestrebt. Als Voraussetzung wird die Vernetzung aller Leistungserbringer und ihrer Informationssysteme sowie die Verankerung des Informationsaustausches auf etablierten Formaten zur semantischen und syntaktischen Vereinheitlichung impliziert, um das informationslogistische Prinzip umzusetzen.

Weitere Voraussetzung für die integrierte Versorgung ist die Änderung der Prozesssicht entsprechend einer Ausrichtung am Patienten und des zugehörigen Behandlungsprozesses, der sich über mehrere Institutionen erstrecken kann. Diese Entwicklung zur digitalen Informationsintegration ist auch geeignet in der medizinischen Dokumentation abzubilden (Berg/Toussaint 2003, 223-224). In dieser Hinsicht erfolgt der Übergang zu einer integrierten, Patienten orientierten Patientenakte meist in mehreren Schritten: Oftmals werden dabei fünf Stufen der zunehmenden Digitalisierung, eines umfangreicher werdenden Integrationsradius sowie einer verstärkten Orientierung am Patienten differenziert (Waegemann 1999, 116):

- *Automated Medical Record*: Vereinzelte Abteilungen arbeiten mit EDV-Systemen.
- *Computerized Medical Record*: Die eingesetzten IT-Systeme werden durch digitale Archive ergänzt.
- *Provider-based Electronic Medical Record*: Das System erlaubt eine vollständige, elektronische medizinische Dokumentation innerhalb einer Institution.
- *Electronic Patient Record*: Medizinische Daten werden über Institutionsgrenzen hinweg zusammengeführt.
- *Electronic Health Record*: Der Zugang zu den medizinischen Daten wird durch den mündigen Bürger kontrolliert, mit der Möglichkeit, selbstständig Daten zu seiner Gesundheitsakte hinzuzufügen.

Die Umsetzung der durchgängigen, integrierten Versorgung kann dazu beitragen, die Qualität im Behandlungsprozess zu steigern, weil alle relevanten Daten für die Leistungserbringer entsprechend dem informationslogistischen Prinzip zur Verfügung gestellt werden und damit medizinische Fehler reduziert werden können. Falls alle Informationen, die für die Entscheidungsfindung erforderlich sind, zum Zeitpunkt der Behandlung verfügbar sind, können weiterhin mögliche Mehrfachuntersuchungen ausgeschlossen werden. Die alleinige Verfügbarkeit der Daten in digitaler Form macht den manuellen Transfer von Informationen zwi-

schen unterschiedlichen Medien noch nicht obsolet: Denn es ist bei Informationen zwischen einer reinen Dokumentensicht und einer Informationssicht zu differenzieren (Schweiger et al. 2006b, 90): Erstere entsteht durch eine einfache Digitalisierung z.B. über die Erfassung von Dokumenten durch einen Scanner. Die Informationssicht hingegen erfordert eine einheitliche syntaktische und semantische Basis, um die Daten automatisiert ggf. in andere Medien zu transferieren und weiterverarbeiten zu können. Damit können potenzielle Übertragungsfehler sowie zusätzlicher Arbeitsaufwand reduziert werden. Insgesamt können mit der integrierten Versorgung als Konsequenz die Reduktion von Kosten und die Verbesserung der Behandlungsqualität erzielt werden.

Von dem Ideal der integrierten Versorgung ist das Gesundheitswesen zum aktuellen Zeitpunkt noch weit entfernt. Vielmehr werden in der im Entstehen befindlichen TI für die eGK (siehe Abschnitt 4.10) erst die wesentlichen Voraussetzungen geschaffen, auf deren Basis eine integrierte Versorgung angestrebt werden kann. Insofern besteht die Möglichkeit, geeignete Rahmenbedingungen für eine durchgängige Behandlung des Patienten zu schaffen.

Die Ausführungen in Abschnitt 4.2 zum Status quo im Gesundheitswesen verdeutlichen, dass die identifizierten Besonderheiten die Realisierung einer integrierten Versorgung auf der Grundlage des informationslogistischen Prinzips erschweren. Die für eine integrierte Versorgung eingesetzten Informationssysteme müssen dabei insbesondere die geänderte Prozesssicht mit der Orientierung der Behandlung am Patienten geeignet unterstützen. Neben der Gewährleistung des informationslogistischen Prinzips kann Informationssystemen auch ein wesentliches Potenzial bei der Reduktion oder Vermeidung von Fehlern in der Medizin attribuiert werden. Ein Überblick über solche Fehler wird im folgenden Abschnitt 4.3 beschrieben.

4.3 Fehler in der Medizin

Fehler können die Behandlungsqualität verringern (Eikel/Delbanco 2003, 634), wenn Fehlentscheidungen getroffen werden, die die Gesundheit des Patienten negativ beeinflussen. Im Folgenden werden deshalb die Ursachen von Fehlern in der Medizin beleuchtet, um das Potenzial zur Fehlerreduktion durch geeignete Informationssysteme herausarbeiten zu können. Ausgangspunkt für die Ausführungen in diesem Abschnitt ist die Darstellung von Fehlern in der Medizin bei *Lenz et al.* (2005, 105).

Die Ursachen von medizinischen Fehlern werden nicht nur in der menschlichen Natur identifiziert, sondern liegen oft auch außerhalb der Einflussmöglichkeiten von Menschen (Leape 1997, 215) im Systemdesign (Volpp/Grande 2003, 851). Darunter zählen z.B. inhärente Systemfehler, die sich im Prozessdesign (Leape 1997, 219) manifestieren können. Um diese potenzielle Fehlerquelle einzuschränken, kann die aktuell im Entstehen befindliche TI für die eGK einen wesentlichen Beitrag leisten, indem Prozesse auf ihre angemessene Implementierung überprüft und ggf. angepasst werden. Dabei sind auch die Prozess unterstützenden Informationssysteme zu berücksichtigen.

Im Gegensatz zu System inhärenten Eigenschaften liegen die Ursachen von so genannten aktiven Fehlern unmittelbar bei Personen (Volpp/Grande 2003, 851). Qualitätsprobleme werden dabei nicht nur durch fehlendes Wissen verursacht, sondern auch durch die Unfähigkeit, das Wissen konsistent und präzise anzuwenden (Kuperman/Gibson 2003, 38). Die Anwendung des Wissens wird durch die zunehmende und zu erfassende Informationsmenge z.B. bei der Diagnose erschwert. Diagnoseprozesse unterliegen dabei einer Vielzahl externer Faktoren, wie z.B. der Verständlichkeit der Aufzeichnungen von Patientenakten (Bhasale 1998, 312).

Auch *Wilson et al.* (1999) stellen fest, dass ein beträchtlicher Umfang von Fehlern auf kognitiven Ursachen beruht, von denen die meisten jedoch vermeidbar wären (Wilson et al. 1999). Eine der möglichen Ursachen für diese Fehler könnte z.B. der nicht ausreichende Gebrauch von IT sein, um die Informationen geeignet aufzubereiten und nach dem logistischen Prinzip zur Verfügung zu stellen (Wilson et al. 1999).

Sind an einem Behandlungsprozess Akteure aus mehreren Institutionen beteiligt, sind weitere Fehlerquellen zu beobachten (Sachverständigenrat für die Konzertierte Aktion im Gesundheitswesen 2003, 146-149): In einer Studie wurden als Fehlerquellen u.a. der Informationstransfer und unzureichende Kommunikation zwischen den Leistungserbringern (Bhasale 1998, 308) identifiziert. Bei der Erfassung der bisherigen Krankheitsgeschichte des Patienten liefert die Patientenakte relevante Informationen (Bhasale 1998, 312), da sie bei einer Institutionen übergreifenden Behandlung das zentrale Medium für den Informationstransfer darstellt. Deshalb kann die Ursache für Zwischenfälle in dem fehlenden Vorliegen der Patientenakte, in der Unvollständigkeit der Akte oder darin liegen, dass sie nicht angemessen durchgesehen wurde (Bhasale 1998, 312). Weiterhin wurde in der Studie von *Bhasale* (1998) festgestellt, dass Verzögerungen bei der Kommunikation auftreten. Diese werden z.B. verursacht durch die inadäquate Kommunikation zwischen Leistungserbringern, bspw. zwischen Krankenhaus und einem niedergelassenen Arzt (Bhasale 1998, 314).

Hier wird deutlich, dass eine ganzheitliche Betrachtung des Behandlungsprozesses noch nicht durchgedrungen ist. Vielmehr agieren Leistungserbringer isoliert und autonom. Zusätzlich fehlen formale und effektive Kommunikationsstrukturen in der Form von geeigneten Kommunikationsmechanismen (Bhasale 1998, 314). Abhilfe schaffen hier adäquate Informationssysteme, die die jeweiligen Leistungserbringer über Sektoren hinweg in der Form von elektronischen Patientenakten (Sachverständigenrat für die Konzertierte Aktion im Gesundheitswesen 2003, 149) mit den relevanten Informationen versorgen und so eine übertragungsbruchfreie Versorgung mit Informationen gewährleisten (siehe auch Bhasale 1998, 314).

Zusammenfassend kann nun konstatiert werden, dass in mehreren Studien festgestellt wurde, dass medizinische Fehler auftreten und oftmals auch vermeidbar wären, wenn geeignete Informationssysteme zur Aufbereitung und damit zur erleichterten Erfassung durch den Leistungserbringer sowie zur Bereitstellung von Informationen nach dem informationslogistischen Prinzip eingesetzt würden. Werden diese relevanten Informationen vollständig, aktuell und mit einem ubiquitären Zugriff bereit gestellt (Varshney 2005, 57-58 und 61), bzw. können sie wegen ihrer Aufbereitung und Filterung in der zur Verfügung stehenden Zeit kognitiv erfasst werden, können fundierte Entscheidungen über den weiteren Verlauf der Behandlung getroffen werden. Fehlen hingegen relevante Informationen oder werden diese nicht vollständig erfasst, können ggf. falsche Entscheidungen getroffen werden, die die Behandlungsqualität negativ beeinflussen können. Dies trifft nicht nur für die sektoreninterne Behandlung, sondern insbesondere auch für die institutionsübergreifende Versorgung zu (Kuilboer et al. 1997).

Als wesentlich für die unzureichende Kommunikation von Patientenakten zwischen Leistungserbringern kann der weit verbreitete Einsatz von Papier basierten Patientenakten bestimmt werden. Der folgende Abschnitt 4.4 befasst sich deshalb dediziert mit den Problemstellungen, die durch dieses Medium verursacht werden.

4.4 Probleme Papier basierter Patientenakten

Patientenakten auf der Basis von Papier besitzen signifikante Nachteile, die in diesem Abschnitt beleuchtet werden:

Obwohl das medizinische und pflegerische Personal einen beträchtlichen Teil seiner Arbeitszeit – dies kann bis zu 50% umfassen (Simoneit 1998, 60) – mit der Dokumentation verbringt, kann doch festgestellt werden, dass die in Papier basierten Patientenakten enthaltenen Informationen mit den folgenden Unzulänglichkeiten behaftet sind:

- Unvollständigkeit[15]
- Nichtverfügbarkeit zum Zeitpunkt der Behandlung (Zimmerman 1978; Tang/Fafchamps/Shortliffe 1994; Tang/LaRosa/Gorden 1999, 249)
- Nicht angemessene Organisation der Informationen (Tang/LaRosa/Gorden 1999, 249)
- Erschwerte Lesbarkeit der Handschrift (Beard/Keck/Peterson 2005, 17)

Die Ausführungen zeigen, dass Papier basierte Patientenakten medienbedingt die Bereitstellung von Informationen nach dem informationslogistischen Prinzip erschweren. Dabei stellt insbesondere die Kommunikation von Patientendaten zwischen Leistungserbringern unterschiedlicher Institutionen wesentliche Herausforderungen dar. Digitale Patientenakten besitzen hingegen ein wesentliches Potenzial bei der Realisierung des informationslogistischen Prinzips, unabhängig davon, ob der Datenaustausch innerhalb oder über die Grenzen von Institutionen hinaus erfolgt. Die Potenziale von digitalen Patientenakten werden im folgenden Abschnitt 4.5 beschrieben.

4.5 Potenziale für digitale Patientenakten

Aus den Ausführungen zu den Problemstellungen von Papier basierten Patientenakten in Abschnitt 4.4 kann geschlossen werden, dass digitale Dokumente diese Defizite beseitigen können: Digitale Dokumente induzieren die Bereitstellung von Informationen nach dem informationslogistischen Prinzip (McDonald 1997, 215), wobei dabei auch der Ort der Informationserfassung und -bereitstellung zu berücksichtigen ist (Anderson 1997, 85; siehe auch Lenz et al. 2005, 110). Insgesamt können digitale Dokumente einen Beitrag zur Reduktion von Fehlern (McDonald/Tierney 1988, 3435-3436; Wyatt/Wright 1998, 1377) bei der Datenerfassung und zur Vollständigkeit der eingetragenen Daten (Anderson 1997, 84) jeweils durch in Informationssystemen implementierte Prüffunktionalitäten liefern (Beard/Keck/Peterson 2005, 17).

Beard et al. (2005) bestätigten die Auswirkungen der Umsetzung des informationslogistischen Prinzips in einer empirischen Untersuchung: Mit der Steigerung der Vollständigkeit von erforderlichen Informationen kann eine Verbesserung in der Behandlungsqualität erzielt werden, weil Entscheidungen auf der Basis von vollständigen Informationen getroffen werden. Weiterhin kann die Arbeitseffizienz durch den Einsatz von aufbereiteten, die relevanten Daten enthaltenden Zusammenfassungen aus elektronischen Patientenakten gesteigert werden (Whiting-O'Keffe et al. 1985; Willard/Connelly/Johnson 1996), womit die kognitive Erfassung der Daten unterstützt wird. Daraus können die Erhöhung der Arbeitseffizienz (Garrett/Hammond/Stead 1986; Anderson 1997, 84) und dementsprechend eine Reduktion von Kosten (McDonald/Tierney 1988, 3434-3435) abgeleitet werden.

Zusammenfassend stellen *Wyatt/Wright* (1998, 1377) fest, dass digitale Patientenakten neben den bereits erwähnten Vorteilen folgende Auswirkungen besitzen können:

- Unterstützung von klinischen Audits und der Forschung

[15] *Tufo/Speidel* (1971), *Dawes* (1972), *Zuckerman* (1975), *Romm* (1981), *Tang/Fafchamps/Shortliffe* (1994).

- Entwicklung von Behandlungspfaden (siehe auch McDonald/Tierney 1988, 3436-3437)
- Vermeidung von Mehrfachuntersuchungen und -therapien

Mit der Digitalisierung von Daten können insgesamt die Defizite Papier basierter Patientenakten beseitigt werden. Durch die Bereitstellung der Informationen in digitalem Format kann die Durchsetzung des informationslogistischen Prinzips als Voraussetzung für die integrierte Versorgung erreicht werden. Außerdem bieten digitale Patientenakten die Möglichkeit der Integration weiterer Anwendungen zur Unterstützung des medizinischen und pflegerischen Personals. Solche Applikationen werden in den folgenden Abschnitten beschrieben.

4.6 Potenziale der IT in der Medizin

Der sinnvolle Einsatz von IT in der Medizin setzt die Digitalisierung von Daten insbesondere der Patientenakte voraus. Je mehr Informationen digital vorliegen, desto vollständiger kann das informationslogistische Prinzip realisiert werden. Deshalb wird von der IT-Unterstützung im Gesundheitswesen ein wesentlicher Mehrwert in der Versorgung erwartet:

Liegen die für Entscheidungen relevanten Informationen entsprechend dem informationslogistischen Prinzip vor, können bspw. Zwischenfälle identifiziert bzw. vermieden werden (Bates et al. 1994, 404 und 408). Die Einführung von IT führte weiterhin zu Effizienzsteigerungen von Prozessen und einer Qualitätssteigerung der erbrachten Leistungen (Kern/Jaron 2003). Es kann gezeigt werden, dass IT im Gesundheitswesen die Häufigkeit von Fehlern reduzieren[16] und damit das Schadensrisiko für Patienten reduzieren (Bates/Gawande 2003, 2526) sowie die Sicherheit erhöhen kann: Beobachtete Verbesserungen im Behandlungsprozess[17] lassen darauf schließen, dass durch die Unterstützung mit geeigneten Systemen der Informationstechnologie die Sicherheit für die Patienten im Behandlungsprozess von klinischen Umgebungen erhöht werden kann (Bates/Gawande 2003, 2531).

Die genannten Auswirkungen von IT setzen eine geeignete Informationssystemlandschaft voraus. Dafür existieren unterschiedliche Ansätze, über die eine Integration von heterogenen Systemen innerhalb einer Institution erfolgen kann und die im folgenden Abschnitt 4.6.1 als Grundlage für die weiteren Ausführungen in Abschnitt 4.6 beschrieben werden.

4.6.1 Entwicklung von Informationssystemen im Gesundheitswesen

Die im Folgenden dargestellten Ausführungen berücksichtigen einerseits Integrationsansätze für Informationssysteme im Gesundheitswesen und andererseits die anzustrebende Orientierung am Patienten (siehe dazu Abschnitt 4.2.7). Diese Beschreibung ist an die Darstellung bei *Prokosch* (2001) angelehnt:

Ursprünglich wurden in der stationären Versorgung in einzelnen Abteilungen Informationssysteme eingeführt, welche aber meist keine Unterstützung für einen Datenaustausch über Schnittstellen leisteten (Prokosch 2001, 371). Als Folge können redundante und inkonsistente Datenbestände beobachtet werden (Prokosch 2001, 371), die über die isolierten Informations-

[16] *Tate/Gardner/Waever* (1990), *Rind et al.* (1994), *Leape et al.* (1995), *Tate/Gardner/Scherting* (1995), *Bates et al.* (1998), *Evans et al.* (1998), *Petersen et al.* (1998), *Bates et al.* (1999), *Bates* (2000), *Rosenfeld et al.* (2000), *Shabot/Lobue/Chen* (2000), *Bates/Gawande* (2003, 2526).

[17] *Rind et al.* (1994), *Tate/Gardner/Scherting* (1995), *Bates et al.* (1998), *Petersen et al* (1998), *Rosenfeld et al.* (2000).

systeme verteilt sind. Mit HL7 (siehe Abschnitt 4.11.2.1.1) als Austauschformat und dedizierten Kommunikations-Servern konnten in einem ersten Schritt administrative Daten ausgetauscht und die Auftragskommunikation zwischen einzelnen Abteilungen unterstützt werden (Prokosch 2001, 372). Letztere ist nicht nur unidirektional als Auftragsanforderung, sondern bidirektional mit der Unterstützung des Rücktransports der ausgewerteten Daten zu verstehen (Prokosch 2001, 372). In der Klassifikation von *Waegemann* (1999, 116) kann eine Institution mit der beschriebenen IT-Infrastruktur als *Automated Medical Record* verstanden werden, wobei für den Datenaustausch eine einrichtungsinterne Patientenidentifikation erforderlich ist (Prokosch 2001, 372-373), um die verteilt vorliegenden Daten geeignet zusammenführen zu können.

Um die Probleme Papier basierter Akten (siehe dazu Abschnitt 4.4) zu reduzieren, werden Informationssysteme in einer weiteren Entwicklungsstufe mit digitalen Archiven ergänzt: Zunächst werden Papierdokumente durch Einscannen digitalisiert, um sie dann über die Informationssysteme mehreren Anwendern zur Verfügung zu stellen (siehe Prokosch 2001, 373). Aus juristischen Gründen werden neben den digitalen Dokumenten meist zusätzliche Archivierungen in der Form von Mikroverfilmungen angelegt, deshalb spricht *Prokosch* (2001, 373) in diesem Fall von hybriden Archiven. In der *Waegemann*-Klassifikation (1999, 116) wird dieser beschriebene Zustand in einer medizinischen Einrichtung als *Computerized Medical Record* bezeichnet (Prokosch 2001, 373).

Werden die bislang beschriebenen Funktionalitäten eines Informationssystems durch die Möglichkeiten der Dokumentation, welche über die medizinische Basisdokumentation zur Angabe von Prozeduren und Diagnosen hinausgeht (Prokosch 2001, 374), ergänzt, wird der Status der *Provider-based Electronic Medical Record* (Waegemann 1999, 116) erreicht (Prokosch 2001, 374). Die dafür erforderliche Infrastruktur zur Integration von heterogenen Informationssystemen kann durch unterschiedliche Architekturvarianten zur Verfügung gestellt werden. Mögliche Integrationsansätze werden in die Kategorien monolithische Ansätze, heterogene und verteilte Konzepte, komponentenbasierte Ansätze sowie Web basierte Ansätze eingeteilt und im Folgenden beschrieben (Prokosch 2001, 374-375; Sunyaev et al. 2006, 31-34):

Bei *monolithischen Ansätzen* (Reichertz 1979) werden Anwendungen aus den jeweiligen Abteilungen über eine gemeinsame Datenbasis (Central database, siehe Abbildung 4.6-1) integriert, um eine Orientierung von Prozessen am Patienten (Haux 2005, 272) zu erreichen. Bei diesem Ansatz werden die folgenden Ebenen betrachtet: Auf der Patientenebene (Patient proximity level) werden diejenigen Maßnahmen fokussiert, die am Patienten durchgeführt werden. In der vertikalen Abteilungsebene (Vertical departmental layer) werden spezialisierte Systeme eingesetzt, die die jeweiligen Anforderungen zur Durchführung der Maßnahmen erfüllen. Diese Systeme besitzen jeweils lokale Datenspeicher. Die horizontale Dienstebene umfasst Dienstleistungen, die über die Grenzen von einzelnen Abteilungen hinaus gehen und durch Informationssysteme unterstützt werden, die über einen lokalen Datenspeicher verfügen. Auch für den Bereich des Krankenhaus-Managements werden drei Ebenen differenziert: Auf Verwaltungsebene (Office system level) werden die jeweiligen Applikationen betrachtet. Die Funktionalitäten werden in vertikalen Abteilungssystemen (Vertical departmental layer) zur Verfügung gestellt, die jeweils über einen lokalen Speicher verfügen. Die horizontale

Dienstebene (Horizontal service layer) umfasst Anwendungen, die diejenigen Funktionalitäten bereitstellen, die über mehrere Abteilungen hinaus verfügbar sein müssen. Der zentrale Kern (Core) dieser Architektur wird durch die Komponenten zentrale Datenbank (Central database), Kommunikationsmechanismen (Central communication) sowie Infrastruktursystem (System) gebildet. Darüber kann ein Zugriff auf zentral gespeicherte Patientendaten realisiert werden und ein durchgängiger Transfer der Patientendaten durch die jeweiligen Applikationen entsprechend dem Behandlungsprozess abgebildet werden.

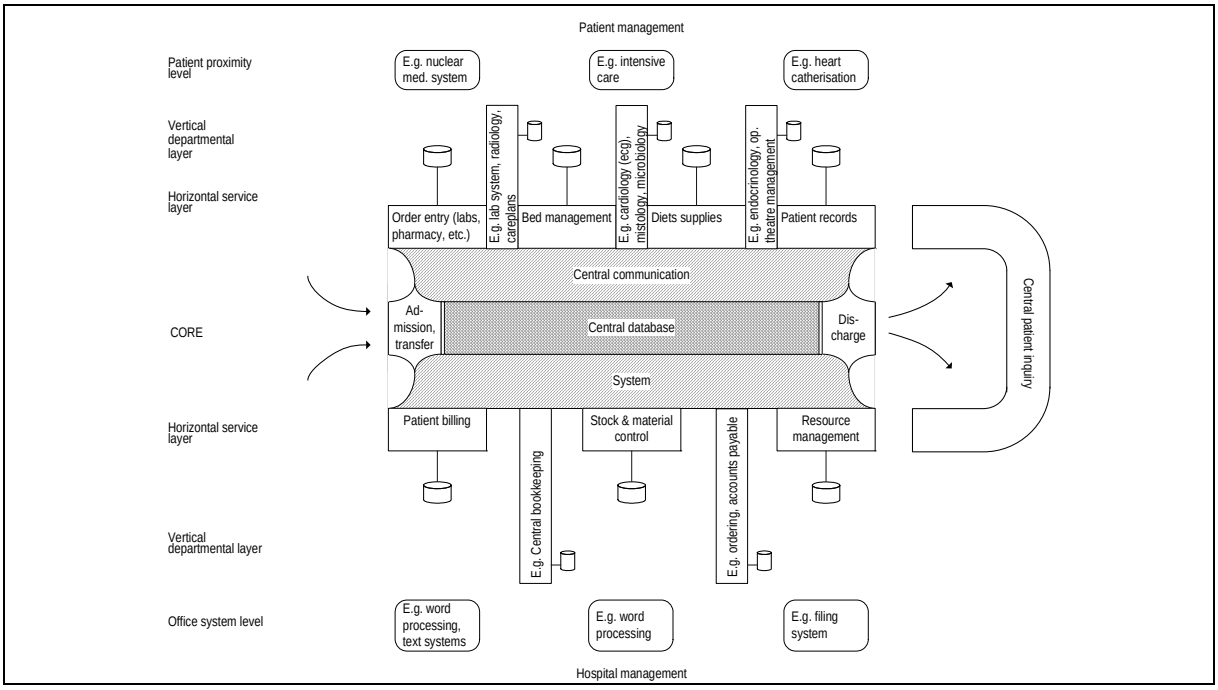

Abbildung 4.6-1: Krankenhausinformationssystem in einem monolithischen Ansatz
Quelle: *Haux* (2005, 272)

Damit verschmelzen die heterogenen Systeme zu einem monolithischen Informationssystem, womit redundante und inkonsistente Daten vermieden und eine Ausrichtung der Informationssystemarchitektur am Patienten erreicht werden. Zusammenfassend kann hinsichtlich einer Kategorisierung auf IS-Architekturebene (Krcmar 2005, 195-196) festgestellt werden, dass der beschriebene monolithische Ansatz integriert und geschlossen ist, weil keine Interoperabilität durch Standards sowie die Beschränkung auf die Integration nur innerhalb einer Institution gegeben sind.

Kommunikationsserver (Gräber 1996; Matthias/Prokosch/Hasselbring 1999; Niemann et al. 2002, 428-429) fungieren bei *heterogenen*, *verteilten Konzepten* (Hasselbring 1997, 196) als Austauschmedium zwischen Spezialanwendungen von Abteilungen, die jeweils über einen lokalen Datenspeicher verfügen. Die Kommunikation zwischen den heterogenen Systemen (z.B. laboratory system, radiology system, siehe Abbildung 4.6-2) erfolgt ausschließlich über diesen Kommunikationsserver (communication server, siehe Abbildung 4.6-2). Aufgabe der Kommunikationsserver ist es, die Nachrichten (messages) der angeschlossenen Systeme zu empfangen, ggf. zwischen unterschiedlichen Protokollen zu übersetzen und an den Empfänger weiterzuleiten (Hasselbring 1997, 195). Da Kommunikationsserver nur auf der Kommunikati-

onsebene fungieren, müssen die angeschlossenen Subsysteme die Adressaten der Nachrichten kennen, die Nachrichten von anderen Subsystemen empfangen und geeignet lokal abspeichern sowie gewährleisten, dass ihre lokalen Datenbestände (database) mit denen der anderen Subsysteme konsistent sind (Hasselbring 1997, 195-196).

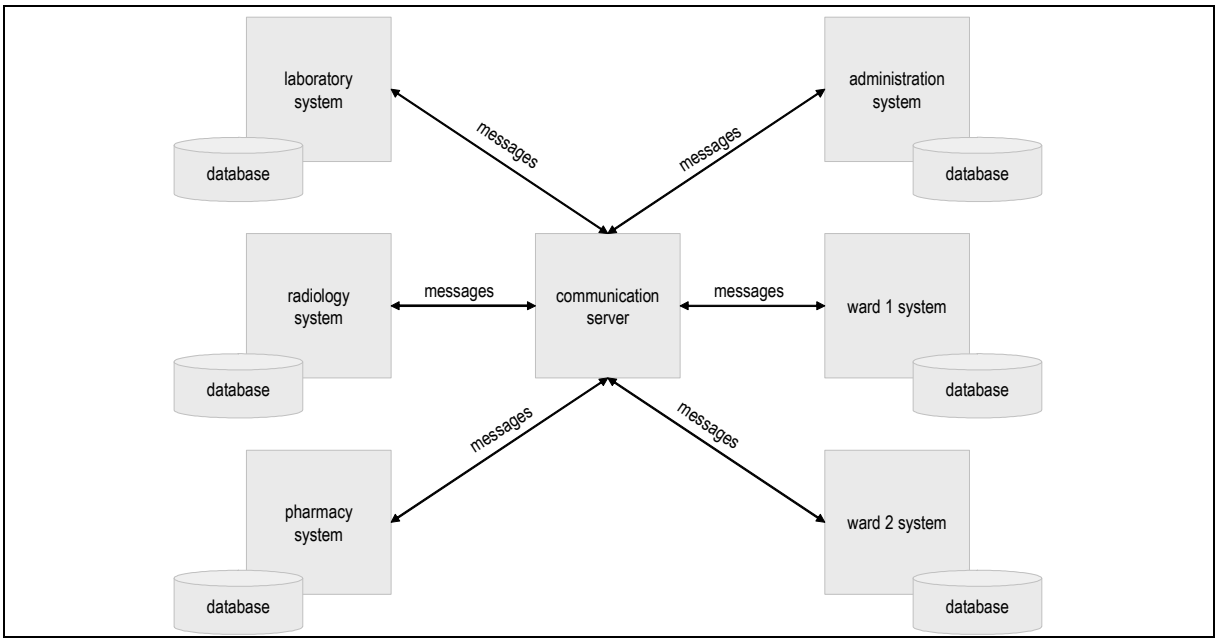

Abbildung 4.6-2: ***Integration heterogener Systeme über einen Kommunikations-Server***
Quelle: *Hasselbring* (1997, 196)

Die Architektur dieser Integrationslösung entspricht dem Prinzip einer Hub-and-Spoke-Topologie im Sinne eines EAI-Ansatzes. Typische EAI-Integrationstechnologien abstrahieren jedoch im Gegensatz zu Ansätzen auf der Basis von Kommunikations-Servern von den tatsächlichen Adressen über den Mechanismus der dynamischen Dienstbindung und -suche. Obwohl die Interoperabilität der Systeme durch den Kommunikationsserver vereinfacht wird, lässt sich zusammenfassend dennoch feststellen, dass der dargestellte Ansatz entsprechend der IS-Architektur-Klassifikation bei *Krcmar* (2005, 195-196) integriert und geschlossen ist, weil nicht notwendigerweise Standards zu berücksichtigen sind und eine Integration nur innerhalb einer Institution angestrebt wird.

Komponentenbasierte Ansätze (z.B. HISA, Healthcare Information Systems Architecture, Ferrara 1997; Scherrer/Spahni 1999; Grimson et al. 2002; zu der der RICHE-Referenzarchitektur folgenden HISA-Spezifikation siehe Frandji 1997) stellen über eine API (Application Programming Interface) Standardfunktionalitäten für ein Krankenhausinformationssystem (siehe Abbildung 4.6-3) anderen Spezialanwendungen zur Verfügung (Prokosch 2001, 374). Ziele sind dabei die Ausstattung der Abteilungen in einem Krankenhaus mit ihren jeweiligen Spezialsystemen sowie die Gewährleistung der Interoperabilität dieser Systeme über einen zentralen und konsistenten Datenbestand (Ferrara 1997, 2). Zur Realisierung der genannten Funktionalitäten wird der ursprüngliche HISA-Ansatz (Ferrara 1997) bei *Scherrer/Spahni* (1999) und *Grimson et al.* (2002) erweitert. Im Folgenden werden auf der Grundlage der Beschreibung des basalen HISA-Konzepts diese Erweiterungen dargestellt:

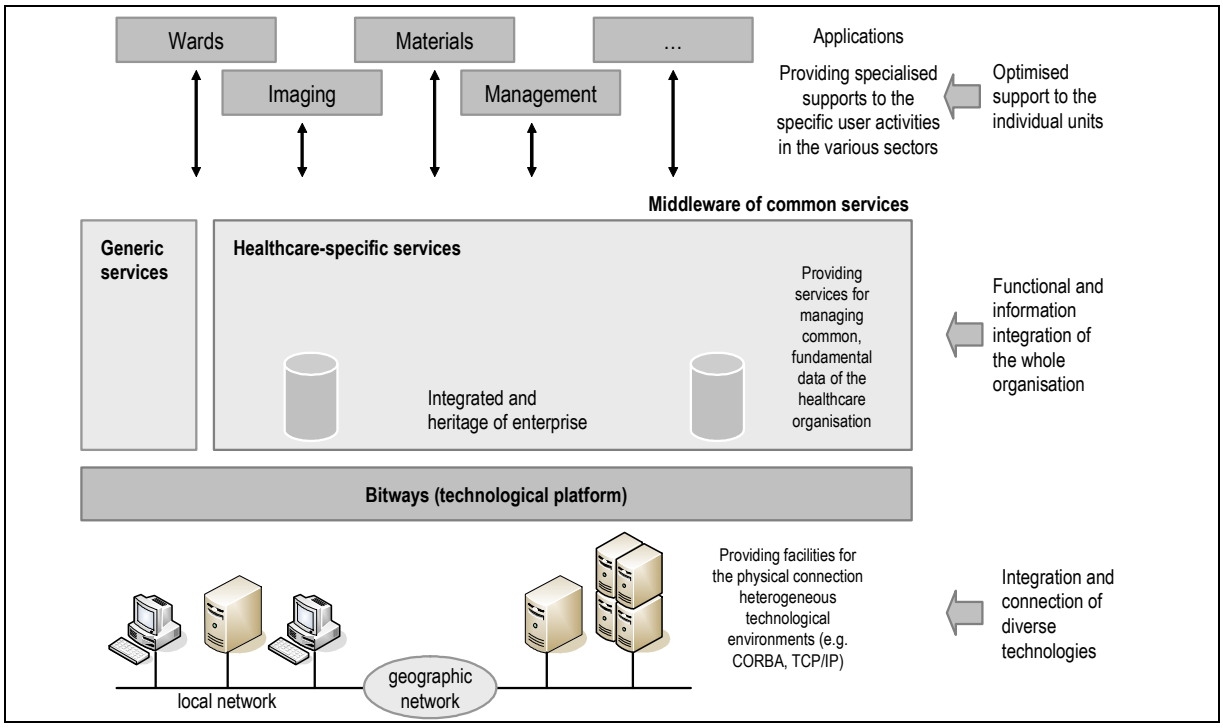

Abbildung 4.6-3: ***Komponentenbasierter Ansatz für ein KIS (HISA)***
Quelle: In Anlehnung an *Ferrara* (1997, 5)

HISA ist in die drei Schichten Applications, Middleware und Bitways strukturiert (Ferrara 1997, 5-6): In der Schicht Applications sind die in den jeweiligen Abteilungen benötigten Spezialsysteme zusammengefasst. Neben dem zentralen Datenspeicher in der Middleware besitzen die Informationssysteme in den Abteilungen jeweils einen lokalen Speicher. In der Middleware werden Dienste implementiert, die diejenigen Daten und Funktionalitäten zur Verfügung stellen, die abteilungsübergreifend erforderlich sind. Dabei wird zwischen generischen (z.B. Verzeichnisdienste, Scherrer/Spahni 1999, 936) und gesundheitsspezifischen Diensten (z.B. Prozessfunktionalitäten, Ressourcenmanagement, Authorisierung, Datenmanagement) differenziert. Die Dienste sind so gestaltet, dass sie an einem zentralen Informationsmodell ausgerichtet sind (Ferrara 1997, 8). Über die Schicht Bitways wird die Interoperabilität auf technischer Basis realisiert.

Der beschriebene HISA-Ansatz wird bei *Scherrer/Spahni* (1999) erweitert, indem die technische Middleware-Funktionalität in die HISA-Middleware integriert wird (siehe Abbildung 4.6-4). Darüber hinaus wird ein weiterer ORB (Object Request Broker) eingeführt (Scherrer/Spahni 1999, 936-937), der von den jeweils eingesetzten Integrationstechnologien wie RPC, DCOM (Distributed COM) oder CORBA abstrahiert und für die Dienste standardisierte Aufrufschnittstellen zur Verfügung stellt (Scherrer/Spahni 1999, 937).

Grimson et al. (2002) erweitern HISA zusätzlich, indem in der Middleware weitere Komponenten eingeführt werden, die auf der Grundlage einer definierten Wissensbasis Client-Applikationen realisieren, die z.B. in der Form von Browsern beim niedergelassenen Arzt über das Internet eine Sichten spezifische Patientenakte zur Verfügung stellen.

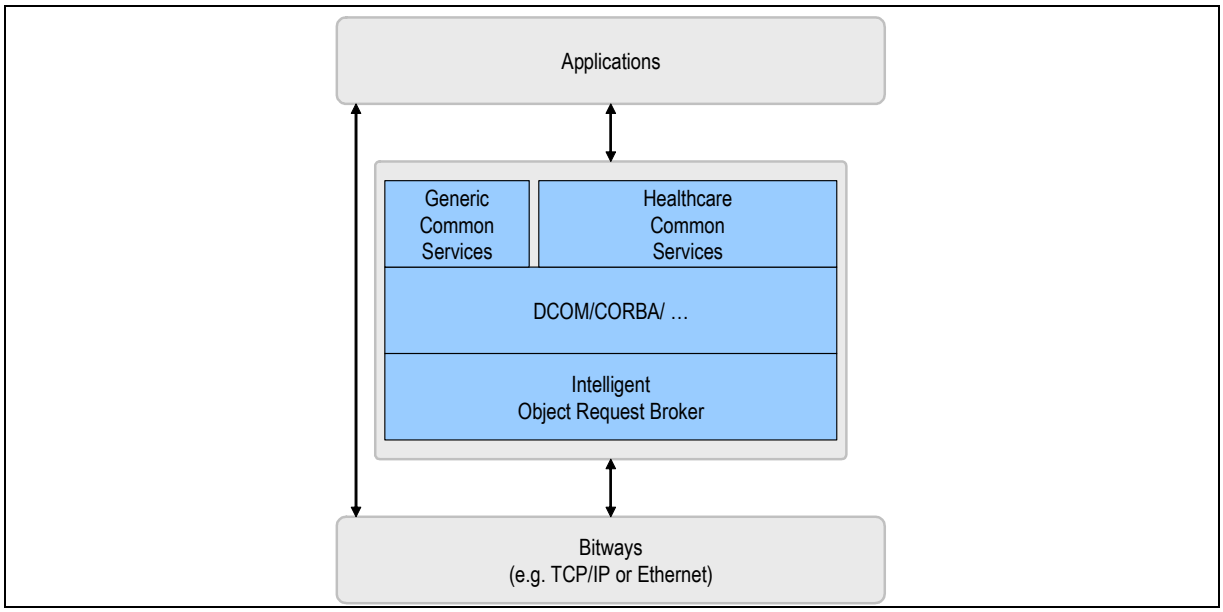

Abbildung 4.6-4: ***HISA mit I-ORB-Erweiterung***
Quelle: In Anlehnung an *Scherrer/Spahni* (1999, 936)

Obwohl *Prokosch* (2001, 374) bei dem bisher beschriebenen Ansatz von einer komponentenorientierten Architektur spricht, ist diese Einordnung nur teilweise gerechtfertigt: Dienste von Komponenten sind zwar über eine wohl definierte Schnittstelle definiert, dennoch entspricht der erweiterte HISA-Ansatz vielmehr einer SOA: Über den I-ORB werden Dienste, die über heterogene Integrationstechnologien implementiert werden, mit wohl definierten Schnittstellen zur Verfügung gestellt. Dabei wird von der jeweiligen Implementierungstechnologie abstrahiert. Die damit angebotenen Dienste orientieren sich hinsichtlich ihrer Funktionalitäten an einem zentralen Datenmodell und stellen damit Dienste im Sinne einer SOA dar. Die Erweiterung zur Aggregation von Sichten spezifischen Daten in einem Browser entspricht der Desktop Integration in einer SOA. Somit sind die wesentlichen Ebenen einer SOA (siehe Abbildung 3.2-6 Abschnitt 3.2.4.2) gegeben, womit der erweiterte HISA-Ansatz als eine SOA betrachtet werden kann. Zusammenfassend lässt sich der HISA-Ansatz entsprechend der IS-Architektur-Klassifikation bei *Krcmar* (2005, 195-196) als offenes und geschlossenes System einordnen, weil über wohl definierte Schnittstellen Funktionalitäten zur Verfügung gestellt werden und eine Integration nur innerhalb der jeweiligen Institution angestrebt wird.

Web basierte Ansätze[18] realisieren ihre Funktionalitäten auf der Grundlage von Internettechnologien, um die Interoperabilität verteilter Informationssysteme zu gewährleisten. Dabei sind insbesondere die durch etablierte Standards induzierte Plattformunabhängigkeit einer Web basierten Applikation (Cimino/Socratous/Clayton 1995, 273; Norris/Dawant/Geissbuhler 1997, 795) sowie der Zugriff auf das Web basierte Informationssystem von beliebigen Lokationen mit einer Internetverbindung (Cimino/Socratous/Clayton 1995, 283) vorteilhaft. Web basierte Ansätze können hinsichtlich ihrer Software-Architektur als Client-Server-Systeme betrachtet werden, wobei im Unterschied zu letzteren zur Kommunikation und Visualisierung etablierte offene Standards eingesetzt und der Datenzugriff von jeder Lokation mit Internet-

[18] *Cimino/Socratous/Clayton* (1995), *Cimino* (1997), , *Norris/Dawant/Geissbuhler* (1997), *Lenz/Kuhn* (2001).

verfügbarkeit unterstützt werden. Insofern können Web basierte Ansätze gemäß der IS-Architektur-Klassifikation bei (Krcmar 2005) als offen und verteilt betrachtet werden.

Die dargestellten Integrationsansätze sind oftmals auf ihre Anwendung innerhalb einer Institution fokussiert. Werden Daten hingegeben nicht nur innerhalb einer einzigen Einrichtung, sondern entlang eines gesamten Behandlungsprozesses über Institutionsgrenzen hinaus benötigt, wird die nach *Waegemann* (1999, 116) benannte Stufe der *Electronic Patient Record* erreicht. Dafür sind die beschriebenen Integrationsansätze jedoch oftmals nicht ausreichend: Die durch die genannten Architekturen realisierte Flexibilität auf den Ebenen der Software- und IS-Architektur ist eingeschränkt: Die betrachteten Integrationsarchitekturen können als die Software-Architekturen monolithisches System, EAI, Komponentenorientierung bzw. SOA und Client-Server-System identifiziert und in die durch sie induzierte IS-Architektur-Klassifikation bei *Krcmar* (2005, 195-196) eingeordnet werden. Die Darstellung in Abschnitt 3.4.2 zeigt, dass diese Architekturen die für Integrationsszenarien relevante Flexibilität auf den Ebenen der Software- und IS-Architektur nur eingeschränkt unterstützen. Insbesondere bei monolithischen Architekturen ist die Kopplung zwischen den heterogenen Systemen per se eng, womit nur eine eingeschränkte Flexibilität gegeben ist.

Der weiter oben in diesem Abschnitt beschriebene Web basierte Ansatz nimmt bei dieser Betrachtung jedoch eine Ausnahmestellung ein: Er bildet als einziger von den in diesem Abschnitt beschriebenen Ansätzen eine Grundlage für eine offene und verteilte IS-Architektur. Damit wird die Informationsintegration über Institutionsgrenzen hinweg (Lenz/Kuhn 2001, 102) durch die Offenheit und Verteiltheit per se unterstützt. Insbesondere kann mit diesem Ansatz über geeignete Konzepte des Information Retrieval (siehe dazu Abschnitt 4.6.6) über Funktionalitäten des Internet das ständig wachsende Wissen in der medizinischen Domäne in aktueller Weise zur Verfügung gestellt werden (Lenz/Kuhn 2001, 102). Insofern verhält sich der Web basierte Ansatz zu den anderen, in diesem Abschnitt beschriebenen Architekturen orthogonal.

Voraussetzung für ein Institutionen übergreifendes Informationssystem ist somit neben einer möglichst standardisierten Kommunikationsinfrastruktur, die die Grundlage für die Interaktion zwischen den einzelnen Einrichtungen bildet, auch eine geeignete Implementierungstechnologie, die die für Integrationsszenarien geforderte Flexibilität möglichst umfassend unterstützt. Die bisher betrachteten Ansätze sind in dieser Dimension meist eingeschränkt. In der vorliegenden Arbeit wird deshalb das Potenzial von agentenbasierten Systemen zur Realisierung von Integrationsarchitekturen beleuchtet.

Die Ausrichtung von Informationssystemen im Gesundheitswesen an Prozessen, die die Versorgung über Institutionen hinaus abbilden, impliziert auch eine entsprechende Fokussierung auf den Patienten: Um das informationelle Selbstbestimmungsrecht der Patienten zu forcieren, muss eine stärkere Orientierung am Patienten erfolgen. Diese Änderung, die durch eine institutionsübergreifende digitale Bereitstellung von Daten erreicht werden kann, bringt auch eine Veränderung der Rollen der beteiligten Akteure mit sich (Eysenbach 2001). *Eysenbach* beschreibt den Begriff der e-health in diesem Zusammenhang wie folgt (2001):

> **„e-health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology".**

Wenn neben Institutionen übergreifenden Daten auch weitere Wellness-Daten, wie z.B. das sportliche Engagement des Patienten, in das Konstrukt *Electronic Patient Record* integriert werden, wird die höchste Stufe *Electronic Health Record* in der *Waegemann*-Klassifikation (1999, 116) erreicht. Als Beispiel für eine solche Plattform, welche als Drehscheibe für den Informationsaustausch zwischen den Leistungserbringern fungieren und die Integration von Patienten unterstützen kann (Prokosch 2001, 378-379), kann der Web basierte Ansatz akteonline.de[19] genannt werden. Insofern erweitern Web basierte Ansätze die Eigenschaft der Orthogonalität zu Konzepten, die lediglich die Integration von Informationen fokussieren, indem der Aspekt der Patientenorientierung ergänzt wird.

Ingesamt kann hinsichtlich der Reichweite der Datenbereitstellung von Informationssystemen im Gesundheitswesen die folgende Entwicklung festgestellt werden: IT-Systeme unterstützen zunächst isolierte Arbeitsprozesse und werden zu innerhalb einer Institution sowie über deren Grenzen hinaus interagierenden Informationssystemen erweitert (Sauer 1999, 282-283; Schoenberg/Safran 2000; Kuhn et al. 2006). Die Realisierung dieser Ziele wird durch eine Implementierung auf der Basis geeigneter Architekturen angestrebt.

Bei der Betrachtung ausgewählter Architekturen zur Informationsintegration im Gesundheitswesen in diesem Abschnitt lässt sich jedoch feststellen, dass diese entweder in diejenigen (nicht agentenbasierten) Architekturen klassifiziert werden können, die in Abschnitt 3.4.2 evaluiert wurden, oder sich als monolithische Anwendung wegen der fehlenden Modularisierung sowie engen Kopplung per se als inflexibel erweisen. Somit sind diese Architekturen nur eingeschränkt geeignet für die Realisierung der bei Integrationsszenarien geforderten Flexibilität. Eine Sonderstellung nehmen dabei Web basierte Ansätze ein, die per se auf der Ebene der IS-Architektur offen und verteilt sind. Somit stellen sie eine geeignete Grundlage für die Integration des Patienten in den Behandlungsprozess, von Informationen zwischen Institutionen, aber insbesondere auch aus etablierten Wissensdatenbanken mit Internet-Konnektivität dar. Insofern und wegen der Integration des Patienten verhalten sich Web basierte Ansätze orthogonal zu den übrigen, in diesem Abschnitt beschriebenen Architekturen.

Insgesamt wird deutlich, dass eine adäquate Kommunikationsinfrastruktur auf der Basis einer geeigneten Implementierungstechnologie für die integrierte Versorgung vorauszusetzen ist. Weiterhin ist Flexibilität von Bedeutung, zu deren Unterstützung sich agentenbasierte Systeme vorteilhaft einsetzen lassen (siehe dazu Abschnitt 3.4.2). Deshalb werden diese in der vorliegenden Arbeit auf ihre Eignung zur Realisierung eines Informationssystems für die integrierte Versorgung beleuchtet.

Auf der Basis einer adäquaten Kommunikationsinfrastruktur können weitere Funktionalitäten zur Unterstützung der Handlungsträger im Behandlungsprozess implementiert werden. Solche Potenziale von IT in der Medizin werden in den folgenden Abschnitten 4.6.2 mit 4.6.8 beschrieben, wobei die Darstellungen in den Abschnitten 4.6.2 mit 4.6.5 die Ausführungen bei Bates/Gawande (2003) zusammenfassen.

4.6.2 Kommunikation, Kooperation und Kommunikation

An der Behandlung eines Patienten – insbesondere auch in einem Krankenhaus – sind meist mehrere Leistungserbringer oft aus unterschiedlichen Disziplinen beteiligt. Die umfangreichen und komplexen Aufgaben, die im Rahmen eines Behandlungsprozesses anfallen, bedür-

[19] Siehe dazu http://www.akteonline.de, zugegriffen am 10.08.2006 und die Arbeiten von *Ückert et al.* (2002), *Ückert et al.* (2003), *Ückert/Müller* (2004), *Schwarze et al.* (2005a).

fen der Abstimmung zwischen den Aufgabenträgern, um deren Effizienz und Effektivität zu gewährleisten. Die Koordination dieser Aufgaben stellt eines der wichtigsten organisatorischen Elemente in der Behandlung dar (Anderson 1997, 85), dennoch kann festgestellt werden, dass zwischen den Handlungsträgern oftmals nur eine Koordination in geringem Umfang beobachtet werden kann (Volpp/Grande 2003, 854).

Informationssysteme können dazu beitragen, die Kommunikation, Kooperation und Koordination zwischen den Akteuren zu verbessern. So können Informationen z.B. bei Schichtübergabe den relevanten Personen zur Verfügung gestellt werden (Volpp/Grande 2003, 852) und damit Zwischenfälle vermeiden helfen (Petersen et al. 1998).

Für eine Steigerung der Effektivität und Effizienz der Interaktionen durch geeignete Informationssysteme sind, wie die Ausführungen in Abschnitt 3.5 zeigen, eine geeignete Modellierung der Interaktionen sowie ihre Unterstützung durch adäquate Implementierungstechnologien erforderlich. Die Darstellungen in diesem Abschnitt zeigen, dass dafür ein wesentliches Potenzial gegeben ist.

4.6.3 Pflichtfelder und Berechnungen bei der Medikation

Informationssysteme können dazu eingesetzt werden, die Eingabe von erforderlichen Daten mit den nötigen Details zu forcieren, wie sie z.B. beim Ausfüllen einer Medikamentenverordnung gegeben ist (Bates/Gawande 2003, 2527). Fehlen bspw. relevante Eingaben, kann der Benutzer nicht zum nächsten Dialogschritt wechseln (siehe auch Bates/Gawande 2003, 2527-2528).

Eine weitere Hilfe stellt die Bereitstellung von Vorschlägen von zusätzlichen Maßnahmen bei der Auswahl eines bestimmten Schrittes im Behandlungsprozess dar (Bates/Gawande 2003, 2528). Automatische Berechungsvorschläge zur Medikation können das Personal weiterhin unterstützen und mögliche Fehler vermeiden helfen (Bates/Gawande 2003, 2528).

In einer Studie (Bates et al. 1998) konnten die in diesem Abschnitt dargestellten Aussagen untermauert werden, indem die signifikante Reduktion von Fehlern im Zusammenhang mit der Medikation beim Einsatz eines geeigneten Informationssystems nachgewiesen werden konnte. Dementsprechend ist zu fordern, adäquate Informationssysteme zur Unterstützung der Handlungsträger bei der Medikation einzusetzen.

4.6.4 Überwachung von Daten

Informationssysteme können das medizinische und pflegerische Personal bei der Überwachung von z.B. automatisch generierten Vitaldaten unterstützen: Es wird dabei oftmals eine Fülle von Daten erzeugt, die das Erkennen von Unregelmäßigkeiten oder Auffälligkeiten erschwert (Bates/Gawande 2003, 2528). Es zeigt sich, dass Menschen mit ihren kognitiven Fähigkeiten in dieser Hinsicht gegenüber einem Informationssystem eingeschränkt sind (Celi et al. 2001). Geeignete Informationssysteme helfen bei der Filterung der Daten (Kuperman et al. 1999, 522), um schließlich eine angemessene Überwachung von relevanten Informationen zu ermöglichen.

Insgesamt können mit adäquaten IT-Systemen eine verbesserte Behandlungsqualität sowie Effektivitäts- und Effizienzsteigerungen erzielt werden, weil erforderliche Maßnahmen einfacher erkannt und damit zeitnah veranlasst werden können.

4.6.5 Entscheidungsunterstützung

Mit geeigneten Informationssystemen kann eine Entscheidungsunterstützung im medizinischen Alltag ermöglicht werden (Anderson 1997, 85). Das informationslogistische Prinzip kann dabei verstärkt werden, indem relevante Informationen in Abhängigkeit von einem veränderlichen Kontext zur Verfügung gestellt werden. Hinsichtlich dieser Kontextsensitivität können unterschiedliche Facetten differenziert werden:

- Ein Bezug zur jeweiligen *Patientensituation* kann einen Mehrwert leisten (Leape 1997, 219), weil damit die relevanten Informationen betrachtet werden.
- Fokussiert auf die unterschiedlichen *Sichten* der am Behandlungsprozess Beteiligten, erleichtern solche Systeme die Bewältigung der inhärent komplexen Eigenschaften des Gesundheitswesens. Berücksichtigt werden als Kontexte z.B. die Ausbildung sowie unterschiedliche Stufen der Anwenderexpertise (Kuziemsky/Jahnke 2005, 23).
- Nomogramme werden in der Medizin sehr häufig in der Entscheidungsunterstützung eingesetzt (Kaparthi/Woodlief 2005, 2), indem die weiteren Behandlungsschritte aufgrund der Erkenntnisse früherer Behandlungsschritte angepasst werden. Als ein wesentliches Element für ein IT basiertes Entscheidungsunterstützungssystem zur Generierung von Nomogrammen wird u.a. die *Integration früherer Daten* angeführt (Kaparthi/Woodlief 2005, 22).

Werden solche Entscheidungsunterstützungssysteme im medizinischen Alltag eingesetzt, können folgende Auswirkungen erwartet werden:

- Verbesserung des Behandlungsergebnisses für den Patienten durch den Zugriff auf relevante Richtlinien für den klinischen Alltag (Anderson 1997, 85; Grimshaw/Russel 1998) und damit eine Reduktion der Aufenthaltsdauer (Adams et al. 1986, 802)
- Verbesserung von Behandlungsprozessen[20]
- Anzeige möglicher Wechselwirkungen mit anderen Medikamenten (Anderson 1997, 85)
- Reduktion von Fehlern durch Erinnerungsfunktionalitäten (McDonald 1976; Leape 1997, 219; Lenz et al. 2005, 110)
- Vereinfachte Identifikation von Diskrepanzen zwischen Soll- und Istwerten bei der Überwachung von medizinischen Daten (McDonald 1976, 1353- 1354).
- Steigerung der Genauigkeit der Diagnosen (Adams et al. 1986, 801).

Die Ausführungen in diesem Abschnitt zeigen, dass Entscheidungsunterstützungssystemen ein wesentlicher Beitrag beim Treffen von Entscheidungen attribuiert werden kann. Entscheidend dabei ist die Berücksichtigung des jeweiligen Kontextes, der wesentlichen Einfluss auf die Qualität, Effektivität und Effizienz der Unterstützung besitzt. Die Arbeit von Wilczek (2007) zeigt jedoch, dass die Integration des jeweiligen Kontextes noch nicht vollständig gewährleistet werden kann. Deshalb sind dafür noch weitere Arbeiten erforderlich, um geeignete Konzepte zur Verfügung zu stellen.

4.6.6 Information Retrieval

Um das medizinische Wissen aktuell zu halten, wird von Leistungserbringen oftmals wissenschaftliche Literatur betrachtet (Sarnikar/Zhao/Gupta 2005, 2554). Um die Informationsflut in

[20] *McDonald* (1976), *McDonald et al.* (1984), *McPhee et al.* (1991), *Rind et al.* (1994), *Pestotnik et al.* (1996), *McDonald* (1997, 215).

den Dimensionen Zunahme der Publikationen und des Umfangs der Patientenakten (siehe auch Simoneit 1998, 92-93) beherrschen zu können, bietet das Konzept des Information Retrieval einen viel versprechenden Ansatz. Dabei liegt der Fokus auf den folgenden Aspekten der Informationsbereitstellung (Wiesman et al. 2006, 25-26):

- *Informationsbedarf*: Im ersten Schritt ist der Informationsbedarf zu identifizieren. Dies ist keine triviale Aufgabe, zumal dem Benutzer der Informationsbedarf unklar und dieser zudem variierend sein kann (Frants/Shapiro/Voiskunskii 1996). Weiterhin besteht oftmals impliziter Informationsbedarf, der dem Anwender so nicht explizit bewusst ist (Wiesman et al. 2006, 25). Im Gegensatz dazu wird unterdrückter Informationsbedarf (Ely et al. 2002) vom Anwender zwar identifiziert, jedoch z.B. aus Zeitgründen nicht erfüllt.
- *Indexierung*: Um den Gehalt eines Dokuments darzustellen, ist dessen Inhalt mit einem Index zu versehen, um die Relevanz für einen bestimmten Informationsbedarf bewerten zu können. Eine Gewichtung der gewählten Schlüsselwörter hilft bei der späteren Auswahl für den Anwender (Wiesman et al. 2006, 25).
- *Matching*: Stimmen der vorab determinierte Informationsbedarf und die Indexierung der Dokumente überein, ist ein potenziell relevantes Dokument identifiziert. Eine zusätzliche Gewichtung der Übereinstimmung ermöglicht die Einordnung in eine Reihenfolge zur Bestimmung der Relevanz der Dokumente (Wiesman et al. 2006, 26).
- *Wissen*: Ziel ist die Reduktion des für das Information Retrieval erforderliche Wissens des Benutzers bei der Interaktion mit dem System. Unterschieden werden dabei zwei Bereiche: Domänenwissen ist dann relevant, wenn die Suche verfeinert werden muss. Helfen können dabei medizinische Thesauri, z.B. UMLS® (United States National Library of Medicine 2005, siehe dazu auch Abschnitt 4.11.3.3). Das Suchwissen hingegen bezieht sich auf das Wissen zur Benutzung des Systems oder der Suchstrategie.
- *Zeit*: Ziel eines Systems mit der Unterstützung des Information Retrieval ist die Reduktion der Suchzeit, die für die Recherche nach bestimmten Informationen zur Behandlung eines Patienten aufzuwenden ist.

Sarnikar et al. (2005) stellen einen Ansatz vor, der die umfangreiche Suche in etablierten Datenbanken nach Fachpublikationen auf die Rolle des Benutzers sowie das jeweilige Interessensgebiet zuschneidet. Dabei wird ein zweistufiger Ansatz verfolgt, der die Informationen filtert, indem inhalts- und regelbasierte Methoden angewendet werden, um die Informationsmenge zu reduzieren und an die Bedürfnisse des Anwenders anzupassen. Die Erstellung eines Benutzerprofils, welches die dafür relevanten Informationen reflektiert, erfolgt bei *Sarnikar et al.* (2005) jedoch manuell, während diese Aufgabe bei *Wiesman et al.* (2006, siehe dazu die Ausführungen weiter unten) durch ein Agentensystem automatisiert wird.

Wiesman et al. (2006) beschreiben den Ansatz des Metabrowsing und ein Agentensystem zur Implementierung des Information Retrieval, um wissenschaftliche Publikationen entsprechend dem Informationsbedarf zur Verfügung zu stellen (siehe Wiesman et al. 2006, 24). Die Ergebnisse dieser Arbeiten werden im Folgenden zusammengefasst (Wiesman et al. 2006, 24, 29 und 31):

Metabrowsing als eine Methode zur Auswahl und Navigation von Domänenkonzepten und Dokumenten fokussiert den Informationsbedarf und das Wissen. Durch die visuell orientierte Methode wird der Benutzer unterstützt, Relationen zwischen den Konzepten und Dokumenten des jeweiligen Anwendungsgebietes zu erstellen.

Der agentenbasierte Ansatz zeichnet sich durch eine im Vergleich mit dem Metabrowsing reduzierte Benutzerinteraktion aus und konzentriert sich auf die Reduktion der Suchzeit und schließt implizite sowie unterdrückte Informationsbedarfe ein. Die Basis für die Suche nach geeigneten wissenschaftlichen Publikationen ist die Integration der Suchmaschine einer medizinischen Datenbank wie z.B. MEDLINE®. Der Informationsbedarf der Anwender wird dabei durch die Interaktion mit einer elektronischen Patientenakte antizipiert. In einer Fallstudie wurde festgestellt, dass der Umfang der abgeleiteten Informationsbedürfnisse so hoch ist, dass das Agentensystem eine zu umfangreiche Menge an Literatur bereitstellen würde, die eine Weiterverarbeitung der Ergebnisse nicht mehr durchführbar gestaltet. Abhilfe kann hier die weitere Filterung der Ergebnismenge darstellen: Einerseits kann die Beachtung der aktuell relevanten Patienteninformationen bezogen auf den momentanen Arztbesuch oder Krankenhausaufenthalt eine Reduktion der Ergebnisse ermöglichen. Die bei *Wiesman et al.* (2006) beschriebene Lösung unterstützt bislang jedoch nicht die Integration in einen Behandlungsprozess. Andererseits wird durch die Integration der jeweiligen Spezialisierung des Personals eine Reduktion der Suchergebnisse erwartet.

Um die Bereitstellung von Information weitergehend zu unterstützen, wird von *Berwick* vorgeschlagen, die relevanten Informationen im Voraus zur Verfügung zu stellen (1996, 272): Dafür ist der Informationsbedarf beim Auftreten von bestimmten Bedingungen zu antizipieren. Beim Auftreten eines definierten Symptoms werden bspw. automatisch relevante Patienteninformationen und klinisches Wissen bereitgestellt. Diese liegen dann beim eigentlichen Untersuchungstermin bereits vor.

Die Ausführungen in diesem Abschnitt zeigen, dass mit dem Information Retrieval die Realisierung des informationslogistischen Prinzips umgesetzt werden kann. Dabei leisten insbesondere agentenbasierte Systeme Mehrwerte gegenüber herkömmlichen Implementierungen. Um die bereitgestellte Informationsmenge zu reduzieren, ist die Beachtung von Kontexten entscheidend. Die Arbeiten von Wilczek (2007) demonstrieren aber, dass dafür noch weiterführende Konzepte zu entwickeln sind.

4.6.7 Digitale Auftragseingabe

Digitale Auftragseingabe ermöglicht den am Behandlungsprozess Beteiligten, Aufträge digital zu erfassen und an den entsprechenden Empfänger zur Weiterverarbeitung zu senden. Mit der Einführung eines solchen Systems konnten in einer Reihe von Studien (Kuperman/Gibson 2003, 32-34) bzw. durch sachkundige Argumentation die folgenden positiven Auswirkungen identifiziert werden (Kuperman/Gibson 2003, 31 und 34-35):

- Verbesserung der Behandlungssicherheit
- Übereinstimmung mit Leitlinien
- Standardisierung von Prozessen
- Verbesserung der Kommunikation zwischen den Abteilungen

- Verbesserung der Effizienz

Zusammenfassend lässt sich feststellen, dass Systeme zur Unterstützung der digitalen Auftragseingabe erlauben, standardisierte Prozesse im klinischen Alltag geeignet zu unterstützen. Besonders vorteilhaft erweisen sich solche Systeme dann, wenn Daten an ihrem Entstehungsort erfasst werden können. Dafür sind mobile Endgeräte prädestiniert, deren vielfältige Einsatzpotenziale im folgenden Abschnitt 4.6.8 beschrieben werden.

4.6.8 Einsatzpotenziale für mobile Endgeräte

Im Krankenhaus kann eine Reihe von Anwendungen identifiziert werden, die sich durch den Einsatz von mobilen Geräten effizienter gestalten lassen und somit einen Mehrwert für Personal und Patienten bieten (siehe für die folgenden Ausführungen auch Schweiger et al. 2006a, 35-36; Hillebrand 2006, 45-55):

- *Befund- und Leistungsdokumentation*: Durch die Einführung der Kostenberechnung auf der Basis von Diagnosis Related Groups (DRG) ist der Dokumentationsaufwand in Krankenhäusern deutlich gestiegen. Um eine DRG richtig zu bestimmen, ist eine vollständige und exakte Erfassung aller gestellten Diagnosen und Nebendiagnosen unabdingbar (Der Bayerische Landesbeauftragte für den Datenschutz 2005). Ohne eine Unterstützung durch mobile Geräte müssen die Daten meist doppelt erfasst werden: zuerst in Form von Notizen auf Formularen und später durch eine manuelle Eingabe in ein IT basiertes Informationssystem. Dies stellt einen zusätzlichen Arbeitsaufwand für das Personal dar und birgt zudem die Gefahr, dass bei der manuellen Datenübertragung Fehler auftreten.
- *Prozessbegleitung bei der Visite*: Durch den Einsatz von mobilen Geräten könnte das Mitführen der Papier basierten Patientenakte bei der Visite überflüssig werden, da alle aktuellen Informationen über die mobile Anwendung abrufbar sind. Daten und Anordnungen könnten außerdem vor Ort erfasst werden (Leimeister et al. 2005, 75) und sind für alle Beteiligten sofort abrufbar (Rügge 2004), wenn die mobile Applikation entsprechende Mechanismen zur Synchronisation mit stationären Anwendungen unterstützt.
- *Unterstützung des Pflegepersonals bei Routinetätigkeiten*: Kontinuierlich erfasste Werte wie z.B. Blutdruck oder Körpertemperatur werden durch Eingaben auf einem mobilen Endgerät direkt in die elektronische Patientenakte integriert (Hillebrand 2006, 47).
- *Unterstützung bei der Terminverwaltung*: Auch beim Festlegen von neuen Terminen können mobile Geräte das Personal unterstützen, indem erstere bereits am Krankenbett vorgemerkt und bestätigt werden (Hillebrand 2006, 49). Zudem könnte das Personal durch entsprechende Funktionen auf bevorstehende Termine ihrer Patienten hingewiesen werden (Lehmann 2005; Hillebrand 2006, 49).
- *Fragebögen zur Erfassung des subjektiven Befindens von Patienten*: Durch das Aushändigen eines PDAs (Personal Digital Assistant) vor der Visite kann mit Hilfe eines elektronischen Fragebogens das Befinden des Patienten ermittelt werden (Walker/Köberle/Strasser 2005). Die Daten sind für den Arzt sofort zugänglich und die Behandlung kann somit optimal auf den Zustand des Patienten abgestimmt werden (Walker/Köberle/Strasser 2005).
- *Integration des Patienten in den Behandlungsprozess*: Werden Patienten mit mobilen Endgeräten ausgestattet, kann eine Verbesserung der Informationslogistik zwischen den einzelnen Akteuren Patienten und Leistungserbringern erzielt werden (Leimeister et al. 2005, 76).

- *Unterstützung bei der Erstellung von Studienprotokollen*: Bei medizinischen Studien können die Probanden durch eine Timer-Funktion regelmäßig an die Dateneingabe erinnert werden. Dadurch erhöhen sich die Genauigkeit sowie Signifikanz der Daten und zudem entfällt die manuelle Auswertung der Fragebögen (Koop/Matesic/Mösges 2001).
- *Visualisierung von medizinischen Sachverhalten*: Mobile Geräte können sinnvoll eingesetzt werden, wenn es darum geht, Patienten komplizierte medizinische Sachverhalte (Hillebrand 2006, 51) mit Hilfe von grafischen Darstellungen und Videoanimationen nahe zu bringen (Bludau/Koop/Herzog 2001).
- *Entscheidungsunterstützung*: Durch die Bereitstellung von kontextbezogenem Wissen kann der Benutzer bei der Entscheidungsfindung unterstützt werden. Mit diesem Ansatz könnten bisher im klinischen Alltag eingesetzte Hilfsmittel (z.B. Tabellenwerke, Handbücher und Taschenrechner, Hillebrand 2006, 53) ersetzt werden (Spreckelsen et al. 2001).
- *Mobile Geräte als Arzneimittelverzeichnis und Medizin-Kompendien*: Auf mobilen Endgeräten können Arzneimittelverzeichnisse Papier basierte Werke ersetzen. Schließlich können die Geräte Lehr- und Nachschlagewerke ersetzen, wodurch sich Informationen an jedem Ort effizient recherchieren und abrufen lassen (Hillebrand 2006, 54).

Mit diesem breiten Anwendungsspektrum wird ersichtlich, dass durch den Einsatz von mobilen Geräten der ständig steigenden Dokumentationspflicht und dem immer größer werdenden Kostendruck entgegengewirkt werden kann: Informationen sind nahezu an jedem Ort und zu jeder Zeit abruf- und erfassbar. Die doppelte Erfassung von Daten wird somit vermieden und eine Menge von Arbeitsabläufen kann effizienter gestaltet werden. Durch die höhere Verfügbarkeit von Daten kann zudem die Behandlungsqualität für den Patienten verbessert werden. Wenn alle relevanten Informationen über ein einheitliches Informationssystem verfügbar sind und zusätzlich auf dieses mit mobilen Endgeräten zugegriffen werden kann, kann weiterhin das Fehlerrisiko für den Patienten reduziert werden (Bates/Gawande 2003, 2527). Neben der Unterstützung des medizinischen und pflegerischen Personals durch mobile Endgeräte können insbesondere mit der Ausstattung von Patienten mit solchen Geräten Verbesserungen in der Informationslogistik und damit Effektivitäts-, Effizienz- und Qualitätssteigerungen im Behandlungsprozess erzielt werden (Leimeister et al. 2005).

Um die genannten Potenziale umsetzen zu können, ist bei der Implementierung von Informationssystemen und der Wahl der eingesetzten Technologien entscheidend, auch eine geeignete, den zur Verfügung stehenden Ressourcen entsprechende Unterstützung von mobilen Endgeräten zu berücksichtigen.

4.6.9 Zusammenfassung zu Potenzialen von IT im Gesundheitswesen

Die in den Abschnitten 4.5 und 4.6.1 mit 4.6.8 beschriebenen Darstellungen zeigen auf, dass Informationssystemen im Gesundheitswesen wesentliche Potenziale für Effektivitäts- und Effizienzsteigerungen sowie Qualitätsverbesserungen von Behandlungsprozessen attribuiert werden können. Eine zusammenfassende Übersicht über die jeweiligen Maßnahmen, angestrebten Ziele und Einordnung in den aktuellen Stand der Forschung werden in Tabelle 4.6-1 beschrieben. Daraus wird ersichtlich, dass in den folgenden Bereichen noch keine vollständigen Lösungen identifiziert werden können:

- Datenintegration über Institutionsgrenzen hinweg mit flexiblen und modularisierten Architekturen
- Fehlende intentionale Modellierung von Information in Informationssystemen
- Fehlende Modellierung und Berücksichtigung von Kontextinformationen
- Unzureichende Berücksichtigung der Ressourcen von mobilen Endgeräten

Die vorliegende Arbeit fokussiert die Integration heterogener Informationssysteme auf der Basis von agentenbasierten Systemen, wobei dabei insbesondere mobile Endgeräte berücksichtigt werden. Kontextinformationen werden jedoch nicht explizit einbezogen. Im folgenden Abschnitt 4.7 werden Defizite bestehender Informationssysteme im Gesundheitswesen beschrieben, die im Widerspruch zu den identifizierten Potenzialen stehen.

Maßnahmen	**Angestrebte Ziele**	**Einordnung in den aktuellen Forschungsstand**
Digitale Patientenakten	• Verbesserung der Informationslogistik • Steigerung der Arbeitseffizienz • Verbesserung der Versorgungsqualität	Meist beschränkt auf Implementierungen innerhalb einer Institution
Integration von heterogenen und verteilten Informationssystemen	• Datenkonsistenz • Reduktion von Datenredundanz • Basis für weitere Anwendungen • Patientenintegration	Etablierte Ansätze besitzen die für Integrationsszenarien geforderte Flexibilität nur in eingeschränktem Umfang.
Unterstützung der Interaktionen zwischen den Handlungsträgern	• Kommunikation • Kooperation • Koordination	Ein intentionaler Charakter der Informationsmodellierung ist oftmals nicht gegeben (Lyyntinen/Lehtinen 1984, 38-39).
Pflichtfelder und Berechnungen bei der Medikation	• Reduktion von Fehlmedikation • Vereinfachung der Medikation	Geeignete Informationssysteme existieren bereits.
Überwachung von Daten	• Filterung von Daten • Steigerung der Effektivität und Effizienz des medizinischen und pflegerischen Personals	Geeignete Informationssysteme sind gegeben.
Entscheidungsunterstützung	• Verbesserung der Behandlungsqualität • Steigerung der Arbeitseffizienz	Eine vollständige Berücksichtigung des jeweiligen Kontextes wird noch nicht unterstützt.
Information Retrieval	Informationsfilterung in wissenschaftlichen Publikationen	Für eine Eingrenzung der Suchergebnisse ist eine Berücksichtigung des Kontextes erforderlich. Dafür existieren noch keine funktional vollständigen Ansätze.
Digitale Auftragseingabe	• Steigerung der Effizienz durch Verbesserung der Kommunikation • Verbesserung der Behandlungsqualität	Geeignete Informationssysteme existieren bereits.
Einsatz mobiler Endgeräte	• Erfassung von Daten an ihrem Entstehungsort • Bereitstellung von Daten am Bedarfsort	Meist unzureichende Berücksichtigung der durch die beschränkten Ressourcen induzierten Eigenschaften von mobilen Endgeräten

Tabelle 4.6-1: ***Maßnahmen, Ziele und Einordnung in den aktuellen Forschungsstand***
Quelle: Eigene Darstellung, zusammengefasst aus den Beschreibungen in den Abschnitten 4.5 und 4.6.1 mit 4.6.8

4.7 Defizite von Informationssystemen im Gesundheitswesen

Trotz der in den Abschnitten 4.5 und 4.6 beschriebenen Potenziale von Informationssystemen kann bei diesen immer noch eine Reihe von Defiziten identifiziert werden. *Beyer et al.* beschreiben Defizite von gegenwärtigen Informationssystemen im Gesundheitswesen zusammenfassend in den folgenden Bereichen (2006, 7-8):

- *Systemintegration*: Für eine Systemintegration sind unterschiedliche Ebenen zu betrachten, die im Folgenden beschrieben werden: Um Systeme zu integrieren, ist eine gemeinsame Datenbasis zu schaffen, die durch eine *Datenintegration* erreicht wird. Zu unterschieden ist dabei zwischen Heterogenität auf der Typebene und der Anwendungsebene. Bei letzterer kann die Einigung auf einen gemeinsamen Wortschatz, wie er z.B. durch UMLS (siehe Abschnitt 4.11.3.3) mit seinen Thesauri festgelegt wird, Abhilfe schaffen. Heterogenität auf Typebene wird durch die Belegung von Daten mit einem bestimmten Typ verursacht. Um diesem Problem zu begegnen, ist eine Einigung auf die Verwendung von gemeinsamen Standards zu erzielen oder die Bindung von Typen an Inhalte so spät wie möglich während der Laufzeit zu erreichen. Mit der *funktionalen Integration* wird die semantische Kopplung der verschiedenen Funktionen aus unterschiedlichen Anwendungen und Komponenten angestrebt. Mit der Systemintegration ist zudem die Benutzerwahrnehmung eines einzigen Systems zu leisten: Für den Benutzer ist eine *Transparenz* zu schaffen, um zu verbergen, mit welchem System der integrierten Landschaft gerade gearbeitet wird. Dies wird zum einen durch ein einmaliges Anmelden an das System und zum anderen durch die automatische Synchronisation der Subsysteme erreicht.
- *Prozessintegration*: Mit dem Ziel der integrierten Versorgung sind Prozesse unterschiedlicher Leistungserbringer und Sektoren zu integrieren. Dies stellt eine der wesentlichsten Herausforderungen bei der Konstruktion von Informationssystemen für das Gesundheitswesen dar (Beyer/Lenz/Kuhn 2006, 8).
- *Domänenevolution*: Da sich die Rahmenbedingungen hinsichtlich des verfügbaren Wissens in der Medizin oder neuer Anforderungen durch veränderte Behandlungsmethoden kontinuierlich verändern (Beyer/Lenz/Kuhn 2006, 8), ist eine flexible Struktur für ein Informationssystem im Gesundheitswesen zu schaffen, das an diese Veränderungen angepasst werden kann und Erweiterungsmöglichkeiten für neue Anforderungen unterstützt.
- *Sicherheit*: Ein Informationssystem mit der durchgängigen Unterstützung des informationslogistischen Prinzips und der Integration des Patienten ist per se ein verteiltes, vernetztes System und unterliegt deshalb Angriffen zur Unterwanderung von Sicherheitseigenschaften. Um die letzteren zu gewährleisten, sind geeignete Mechanismen umzusetzen.

Die Beseitigung dieser Defizite ist das Ziel von nationalen und internationalen Bemühungen. *Lenz et al.* (2005, 111-116) beschreiben entsprechende Anforderungen, die als Voraussetzungen für geeignete Informationssysteme identifiziert und in den Abschnitten 4.8 mit 4.9 detailliert dargestellt werden. Dabei kann zwischen Voraussetzungen für die Etablierung von Informationssystemen im Allgemeinen und für die Informationsintegration im Besonderen differenziert werden.

4.8 Allgemeine Voraussetzungen für medizinische Informationssysteme

4.8.1 Akzeptanz

Der den gewünschten Nutzen reflektierende Einsatz von Informationssystemen setzt die Akzeptanz bei den Anwendern voraus (Bates/Gawande 2003, 2533). Deshalb wird bei *Schwabe/Krcmar* (1996b; 1996a) und *Schwabe* (2001) ein Ansatz (NDA, siehe Abschnitt 1.6) vorgestellt, der die Bedarfsanalyse für Telekooperationsplattformen beschreibt, um auf der Basis der Bedarfsorientierung die Akzeptanz der Anwender (für eine Strategie zur Auswahl relevanter Akteure zur Steigerung der Akzeptanz siehe die Ausführungen bei Mantzana et al. 2007) zu erhöhen. Letztendlich handelt es sich auch bei einer Patientenakte um eine Kooperationsplattform (Krcmar/Horn 2001b, 21), da die am Behandlungsprozess Beteiligten ihre Koordination, Kooperation und Kommunikation über dieses Medium abstimmen. Deshalb ist die Anwendung des NDA auf die Gestaltung einer elektronischen Patientenakte als Interaktionsmedium gerechtfertigt. Daher wird der NDA auch im Rahmen der vorliegenden Arbeit als Basis für die Erarbeitung der Ergebnisse eingesetzt.

Um die Akzeptanz speziell für ein medizinisches Informationssystem zu erhöhen, wird, zusammengefasst aus den detaillierten und auf die Domäne Gesundheitswesen bezogenen Darstellungen bei *Anderson* (1997, 90) und *Bates et al.* (2003, 524-528), vorgeschlagen, die folgenden Aspekte zu berücksichtigen (siehe dazu auch Leimeister et al. 2005, 84):

- Soziale Aspekte
- Organisatorische Perspektive
- Technische Eigenschaften

Letztendlich entscheidet die Akzeptanz der Anwender über die positiven Auswirkungen von Informationstechnologie. Deshalb sind beim Einsatz von Informationssystemen die dafür erforderlichen Voraussetzungen zu schaffen (Garfield/Kamis/LeRouge 2004; Chan/Dekker/Ramsden 2005, 16 und 21-22). Die vorliegende Arbeit fokussiert technische Eigenschaften. Diese werden in den folgenden Abschnitten 4.8.2 sowie 4.9 beschrieben.

4.8.2 Flexibilität

Die Anforderung der Flexibilität kann hinsichtlich zweier Dimensionen betrachtet werden: Einerseits ist die Beherrschung der Flexibilität zur Implementierungszeit erforderlich, indem Informationssysteme entsprechend den Anforderungen konfiguriert werden (siehe auch Lenz et al. 2005, 111). Das Gesundheitswesen zeichnet sich durch eine ständig zunehmende Menge an verfügbarem Wissen aus. Dieses Wissen wird kontinuierlich durch aktuelle Forschungsergebnisse erweitert und in Teilen auch revidiert. Diese internen Treiber für Veränderung werden ergänzt durch externe Treiber, die z.B. in der organisatorischen Veränderung manifest werden, die durch die politischen Vorgaben obligatorisch sind (Lenz et al. 2005, 110). Wesentliche Veränderungen im Gesundheitswesen werden z.B. durch die verbindliche Einführung der eGK oder der Fallpauschalen basierten Abrechnung in Krankenhäusern ausgelöst (siehe auch Lenz et al. 2005, 111). Die zuletzt genannten Institutionen werden dabei insgesamt zu „lernenden“ Systemen (Clade 2003, A2681), welche sich den neuen Herausforderungen stellen müssen (Lenz et al. 2005, 111).

Andererseits müssen Informationssysteme die Dynamik, den Nichtdeterminismus und nicht antizipierbare Ereignisse zur Laufzeit eines Systems beherrschen (siehe dazu auch die Ausführungen in Abschnitt 4.2.2.2 zum Aspekt der Dynamik). Insbesondere für diesen Aspekt der Flexibilität sind bisherige Informationssysteme nur eingeschränkt geeignet.

Bisher verfügbare Informationssysteme unterstützen die zur Implementierungs- und Laufzeit geforderte Flexibilität meist nur teilweise bzw. unzureichend. Um die geforderte Flexibilität zu erfüllen, sind deshalb Informationssysteme im Gesundheitswesen auf der Basis von geeigneten Implementierungstechnologien zu realisieren.

Für die Informationsintegration über Institutionsgrenzen hinweg sind neben den in Abschnitt 4.8 genannten allgemeinen Anforderungen von Informationssystemen besondere Eigenschaften zu erfüllen, die im folgenden Abschnitt 4.9 beschrieben werden.

4.9 Besondere Voraussetzungen für institutionsübergreifende Informationssysteme

Obwohl hinsichtlich der Integrationsproblematik sowohl innerhalb von Institutionen als auch über deren Grenzen hinweg Industrieparallelen identifiziert werden können (Schweiger/Leimeister/Krcmar 2005a, 2005b, 2007) und diese bereits gelöst wurden, ist die medizinische Domäne von der Idealumsetzung eines übergreifenden Informationssystems (Stufe der Electronic Health Record bei Waegemann 1999) noch weit entfernt. Erschwerend wirken sich dabei die besonderen Eigenschaften des Gesundheitswesens aus (siehe dazu die Ausführungen in Abschnitt 4.2.2).

Für den zur Datenintegration erforderlichen Austausch von Informationen zwischen den Leistungserbringern kann die elektronische Patientenakte einen wesentlichen Beitrag leisten. *Prokosch et al.* beschreiben die Anforderungen an eine solche Patientenakte zum institutionsübergreifenden Austausch von Informationen wie folgt (2006, 17):

- *Strukturierte Dokumentation* der grundlegenden Datensätze und *Verknüpfung* aller für eine Behandlungsepisode relevanten Informationen
- Unterstützung des *logistischen Arbeitsflusses* zwischen den beteiligten Sektoren
- Sichere und *verschlüsselte Kommunikation* sowie *Speicherung* von Patientendaten
- Realisierung eines benutzerfreundlichen und flexiblen *Authentifizierungsmechanismus* für den Zugriff auf Daten für diejenigen, die im Behandlungsprozess involviert sind
- Jeder Gesundheitsdienstleister verwendet weiterhin seine eigenen Spezialsysteme. Deshalb ist eine automatisierte *Übertragung und Integration von Informationen* in das jeweilige System des Leistungserbringers erforderlich.
- Etablierung eines gemeinsamen *Standards* für die Struktur der ausgetauschten Dokumente
- Eine national eindeutige *Patientenidentifikation* ist für jeden Patienten zu unterstützen.
- Die *Dateneinträge* in die elektronische Akte müssen strukturiert sein.
- Der *Zugriff* auf Patientendaten nur dann, wenn eGK und Heilberufsausweis gleichzeitig im Kartenlesegerät vorhanden sind, reicht nicht aus.

Als Ansätze für Akten, die die institutionsübergreifende Informationsintegration anstreben, können im deutschen Sprachraum folgende Projekte identifiziert werden (Prokosch et al. 2006, 19):

- akteonline.de[21]
- avetana (Gmelin et al. 2001; avetana GmbH 2005)
- careon (careon.de GmbH 2006)
- LifeSensor (InterComponentWare AG 2001, 2006)
- Fallakte.de: Bidirektionaler Datentransfer zwischen niedergelassenem Arzt und Klinik

Von diesen Ansätzen genügt nur die Web basierte Implementierung akteonline.de dem Anspruch, Informationen zwischen mehreren Leistungserbringern unterschiedlicher Institutionen auszutauschen (Prokosch et al. 2006, 19). Dabei wird die gemeinsame Dokumentenstruktur der Clinical Document Architecture (Dolin et al. 2001) zur Integration von häufig vorkommenden Standards wie HL7 (Health Level Seven Inc. Ann Arbor MI 2005, siehe Abschnitt 4.11.2.1.1) oder DICOM®[22] (siehe Abschnitt 4.11.2.1.2) eingesetzt (Ückert/Müller 2004). Die Arbeit wird auf die Konstruktion eines zentralen, Web basierten Informationssystems beschränkt (Ückert/Müller 2004), ohne jedoch weitere Funktionalitäten wie z.B. die Prozesssteuerung in den Ansatz zu integrieren. Außerdem ist die Modularisierung und Flexibilität wegen des auf dem Client-Server-Modell basierenden Web basierten Ansatzes per se eingeschränkt (siehe dazu Abschnitte 3.4.2.1 und 3.4.2.6).

Neben dem Austausch von medizinischen Daten könnten durch eine umfassendere Informationsintegration auch Verwaltungsdaten übertragen werden, z.B. zur lückenlosen Behandlung des Patienten durch geeignete Anschlusstermine beim nächsten Leistungserbringer (Prokosch et al. 2006, 17). Damit werden nicht nur medizinische Daten ausgetauscht, sondern es wird eine organisatorische Unterstützung des Patienten bei seinen Behandlungen ermöglicht, die sich über mehrere Institutionen erstrecken können.

Insgesamt lässt sich festhalten, dass eine Reihe von Ansätzen zur Informationsintegration existiert. Dabei werden jedoch die besonderen Eigenschaften der Flexibilität zur Laufzeit nicht vollständig unterstützt. Deshalb wird in der vorliegenden Arbeit beleuchtet, welche Eigenschaften eine geeignete Architektur besitzen sollte, die diesem Flexibilitätsaspekt genügt.

In den folgenden Abschnitten 4.9.1 mit 4.9.4 werden die besonderen Anforderungen an ein geeignetes Informationssystem zur Informationsintegration konkretisiert.

4.9.1 Berücksichtigung von Vertraulichkeit

Die Vertraulichkeit bei der Informationsintegration ist aus den folgenden beiden Gründen zu gewährleisten:

[21] *Ückert et al.* (2002), *Ückert et al.* (2003), *Ückert/Müller* (2004), *Gesakon GmbH* (2005), *Schwarze et al.* (2005b).

[22] DICOM is the registered trademark of the National Electrical Manufacturers Association for its standards publications relating to digital communications of medical information.

- Die Steuerung des Gesundheitswesens (siehe dazu auch Abschnitt 1.1) ist abhängig von erhobenen Datensätzen. Diese sind vor ihrer Analyse zu anonymisieren.
- Zugriffsschutz auf die digitalen Patientendaten

Um epidemiologische Vorhersagen treffen zu können und das Gesundheitswesen geeignet steuern zu können, ist die Analyse von medizinischen Daten erforderlich. Hier entsteht aber das Dilemma, dass bisher keine befriedigenden Methoden bestehen, diese Daten von personenbezogenen, geographischen Informationen zu abstrahieren und Schlüsse aus medizinischen Daten zu ziehen (Kamel Boulos et al. 2006, 160). Mit der von *Kamel Boulos et al.* beschriebenen, auf Software-Agenten basierenden Methode (2006) kann ein bedeutender Lösungsansatz realisiert werden, indem aussagekräftige Datenanalysen durchgeführt werden können, ohne jedoch personenbezogene Daten offen zu legen (siehe dazu auch Abschnitt 2.3.10).

Um die Akzeptanz einer integrierten Patientenakte beim Patienten zu erreichen, ist es erforderlich, geeignete Sicherheitsmaßnahmen bei vernetzten Informationssystemen zu etablieren, denn letztere sind besonderen Gefahren (siehe dazu Abschnitt 2.3.10) ausgesetzt. Für eine Akzeptanz sind die Sicherheitsmaßnahmen nicht nur zu implementieren, sondern so zu evaluieren, dass die Ergebnisse intersubjektiv nachvollziehbar sind. Dazu wird von *Sunyaev et al.* (2007) ein geeignetes Konzept als Evaluierungsgrundlage vorgeschlagen. *Sunyaev* (2005) beleuchtet darüber hinaus ausgewählte Sicherheitseigenschaften von Telematikanwendungen im Gesundheitswesen, indem diese auf der Basis der UML-Erweiterung UMLsec (Jürjens 2005) formal bewiesen werden. Diese Maßnahmen zielen auf die Stärkung der Akzeptanz der entwickelten Telematik-Lösungen ab.

Weil der Erfolg von Telematik-Anwendungen im medizinischen Bereich maßgeblich von der Akzeptanz der Anwender abhängt, sind geeignete Maßnahmen zur Gewährleistung der Vertraulichkeit und der intersubjektiven Nachvollziehbarkeit der implementierten Sicherheitseigenschaften zu ergreifen.

4.9.2 Patientenorientierung

Der Patientenorientierung können unterschiedliche Facetten attribuiert werden. Letztere werden in diesem Abschnitt detailliert beschrieben, um ihre Evaluierung zu ermöglichen. Dabei ist zu beleuchten, ob diese Aspekte dem Anspruch der Patientenorientierung gerecht werden und dementsprechend durch technische Maßnahmen zu realisieren sind. Als Ausgangspunkt werden die folgenden Ausprägungen der Patientenorientierung differenziert:

- Integration des Patienten in den Behandlungsprozess
- Übertragung der Datenhoheit an den Patienten
- Eigene Dokumentation durch den Patienten
- Qualitätssicherung von im Web verfügbaren medizinischen Informationen

Die potenzielle Notwendigkeit für die Integration des Patienten in den Behandlungsprozess kann aus einer Reihe von Studien abgeleitet werden, deren Ergebnisse im Folgenden dargestellt werden:

- Patienten besitzen *Interesse* an der Begutachtung ihrer Patientenakten[23].
- Der Zugang für Patienten zu ihren Akten kann ihr *Verständnis* für die Sachverhalte fördern (Laugharne/Stafford 1996, 340).
- Identifiziertes Potenzial für eine verbesserte *Compliance* im Behandlungsprozess (Giglio et al. 1978; Essex/Doig/Renshaw 1990)
- Potenzielle *Verbesserung des Behandlungsprozesses* (Ball/Lillis 2001; Cimino/Patel/Kushniruk 2001; Schwarze et al. 2005b, 193)

Aus den Ergebnissen der aufgeführten Studien kann abgeleitet werden, dass die Integration des Patienten in den Behandlungsprozess wesentlichen Einfluss auf die erzielbare Qualität besitzt. Somit ist eine solche Beteiligung des Patienten durch geeignete technische Maßnahmen zu unterstützen.

Neben der Integration des Patienten in den Behandlungsprozess ist die Übertragung der Datenhoheit an den Patienten erforderlich, um der neuen Patientenrolle eines aktiven Teilnehmers im Behandlungsprozess gerecht zu werden (Brennan/Safran 2003) und die informationelle Selbstbestimmung zu stärken: Im Zusammenhang mit der TI für die eGK bestätigt der Patient z.B. mit der Eingabe seiner PIN (Persönliche Identifikationsnummer) für die eGK dem Leistungserbringer dessen Zugriff auf die darauf gespeicherten bzw. referenzierten Daten und erhält somit die Steuerung zur Gewährung von Zugriffen auf seine Daten. Dementsprechend ist eine adäquate technische Realisierung angemessen, um der neuen Rolle des Patienten gerecht zu werden.

Web basierte Patientenakte erlauben dem Patienten nach *Sittig* (2001, 3) und *Schneider* (2001, N200), eigene Daten einzugeben. Es ist nun kritisch zu reflektieren, ob die Bereitstellung solcher Funktionalitäten sinnvoll ist. Dazu werden die Ergebnisse einer Reihe von Studien beleuchtet:

- Web basierte Patientenakten sind technisch unausgereift (Sittig 2001, 4; Kim/Johnson 2002, 178-179)
- Unzureichende Maßnahmen zur Qualitätssicherung der gespeicherten Daten (Schneider 2001, N197-N198 und N200; Kim/Johnson 2002, 178)

Angesichts dieser Ergebnisse hinterfragen *Kim/Johnson* (2002) nicht ohne Grund kritisch, ob mit den identifizierten Schwächen und Einschränkungen Web basierte Patientenakten, die von Patienten selbst gepflegt werden können, überhaupt dazu geeignet sind, als ein geeignetes Medium zur Unterstützung des Behandlungsprozesses zu fungieren (Kim/Johnson 2002, 178). Obwohl die in den genannten Studien beleuchteten Systeme bislang noch nicht für einen problemlosen Einsatz geeignet sind, ist dennoch eine zunehmende Nachfrage von Patienten nach solchen Systemen zu verzeichnen (Schneider 2001, N199). Dieses Interesse ist aus der Nachfrage von Patienten nach einer qualitativ hochwertigen Gesundheitsversorgung ableitbar (Schneider 2001, N199). Diese Diskrepanz gilt es nun geeignet aufzulösen:

[23] *Baldry et al.* (1986), *Bird/Walji* (1998, 595), *Masys et al.* (2002, 188 und 189), *Fowles et al.* (2004), *Pyper et al.* (2004), *Ross et al.* (2005).

Da bisher verfügbare Lösungen die Voraussetzungen für eine geeignete Integration des Patienten und Qualitätssicherung noch nicht unterstützen, wird vorgeschlagen, von der Implementierung einer vollständig funktionalen, Web basierten Patientenakte abzusehen. Stattdessen sollen durch den Patienten nur solche Informationen eingegeben werden, die aus medizinischer Sicht keine weite Tragweite für Entscheidungen besitzen. Darunter fallen bspw. Daten zur Beschreibung der sportlichen Aktivität des Patienten. Mit diesem Kompromiss wird das Interesse des Patienten an der Integration in den Behandlungsprozess berücksichtigt, indem Daten über eine Web basierte Anwendung zur Verfügung gestellt werden und eine begrenzte Möglichkeit gegeben wird, diese Daten selbstständig zu ergänzen. Außerdem wird die Qualität der in einer Web basierten Patientenakte vorgehaltenen Daten gewährleistet.

In der EU-Ministererklärung vom 22.05.2003 (Ministererklärung Brüssel 2003-05-22 2003) wird betont, dass die Patientenzentrierung durch die Bereitstellung von geeigneten Internet basierten Informationssystemen zu ergänzen ist (Ministererklärung Brüssel 2003-05-22 2003, 2). Weiterhin wird dort für die Gewährleistung der Validität der im Internet hinterlegten Informationen vorgeschlagen, die verfügbaren Inhalte entsprechend zertifizieren zu lassen. Ist diese Qualitätssicherung gegeben, erscheint es sinnvoll, solche Informationen zur Verfügung zu stellen, um dem Patienten die Möglichkeit einzuräumen, sich über seine Behandlung zu informieren und damit seine aktive Mitwirkung am Behandlungsprozess zu stärken.

Mit dem Begriff e-health werden nach Eysenbach (2001) einerseits die technologischen Möglichkeiten des Internets und andererseits eine sich ändernde Beziehung zwischen Arzt und Patient subsumiert (Ball/Lillis 2001). Die in diesem Abschnitt identifizierten Facetten der Patientenorientierung lassen sich in diese beiden Aspekte einordnen.

Bei der technischen Umsetzung der Ausprägungen der Patientenorientierung ist darauf zu achten, dass die gewünschten Effekte wirklich erreicht werden können. Denn bisherige Studien über Web basierte Patientenakten, die von Patienten selbst geführt werden können, zeigen noch deutliche Mängel, die es nicht anraten, zum aktuellen Zeitpunkt solche Patientenakten zur Verfügung zu stellen. Vielmehr ist es erforderlich, einen geeigneten Kompromiss zwischen der von Patienten gewünschten Integration in den Behandlungsprozess sowie einer Qualitätssicherung der selbstständig eingegebenen Daten zu gewährleisten.

4.9.3 Prozessunterstützung

In der ganzheitlich angestrebten Informationsintegration ist die Informationsversorgung insbesondere über Institutionsgrenzen hinweg zu gewährleisten. Die wichtigste Verbindungslinie zwischen stationärer und ambulanter Versorgung ist bislang der Informationsaustausch über Entlassbriefe, deren Hauptfunktion im Datenaustausch zwischen den genannten Versorgungseinrichtungen besteht (van Walraven/Weinberg 1995, 1438). Weil diese Informationskette essentiell für eine qualitativ hochwertige Weiterbehandlung (Moore et al. 2001, 646) ist, ist darauf zu achten, dass die erforderlichen Informationen auch beim adressierten Leistungserbringer im ambulanten Bereich lückenlos ankommen.

In der von *van Walraven/Weinberg* (1995) durchgeführten Studie konnten jedoch erhebliche Mängel bei diesem Informationstransfer identifiziert werden: Bei Entlassbriefen wurden unvollständige Informationen identifiziert (van Walraven/Weinberg 1995, 1440). Weiterhin konnte festgestellt werden, dass etwa die Hälfte der Entlassbriefe den niedergelassenen Arzt nicht erreichten (van Walraven/Weinberg 1995, 1441). Bezüglich der Vollständigkeit und der Effizienz von Entlassbriefen stellen *van Walraven/Weinberg* (1995, 1441) somit beträchtliche

Defizite fest. Wegen der Einschränkung einer nicht vollständig repräsentativen Untersuchung bei *van Walraven/Weinberg* (1995) ist anzumerken, dass die gefundenen Ergebnisse bei einer Übertragung auf allgemeine Aussagen zu relativieren sind. Die grundlegenden Aussagen, dass Entlassbriefe nicht alle erforderlichen Informationen enthalten (Wilson et al. 2001, 105) und nicht jeder Entlassbrief den adressierten niedergelassenen Arzt erreicht, konnten in einer weiteren Studie von *Wilson et al.* (2001) bestätigt werden. Insofern werden die Aussagen über die Unvollständigkeit und Verfügbarkeit von Entlassbriefen beim niedergelassenen Arzt untermauert.

Moore et al. (2001) untersuchten in ihrer Studie, ob nun ein Zusammenhang zwischen der sektorbedingten Diskontinuität zwischen stationärer und ambulanter Versorgung, wie sie in den Studien von *van Walraven/Weinberg* (1995) und *Wilson et al.* (2001) identifiziert wurde, und dem Auftreten von medizinischen Fehlern besteht. Dabei konnte als Ergebnis festgestellt werden, dass ein Zusammenhang zwischen der diskontinuierlichen Versorgung und dem Auftreten von medizinischen Fehlern besteht (Moore et al. 2001, 648), der wiederum in Relation mit einem erhöhten Risiko zur Rehospitalisierung stehen kann (Moore et al. 2001, 650). Zu den gefundenen Ergebnissen ist anzumerken, dass auch hier eine nicht vollständig repräsentative Untersuchung die Relativierung der Resultate bei der Übertragung auf allgemeingültige Aussagen erfordert. Insgesamt ist aber erkennbar, dass der kontinuierliche Informationsfluss für eine durchgängige und qualitativ hochwertige Versorgung relevant ist.

Wilson et al. (2001) kommen wegen der gefundenen Ergebnisse in ihrer Studie zu dem Schluss, dass die Kommunikation zwischen Krankenhäusern und Niedergelassenen in der jetzigen Form zu ändern sei. Um den Kommunikationsfluss zu verbessern, werden zwei Alternativen vorgeschlagen (Wilson et al. 2001, 107): Einerseits kann durch eine kurze und automatische, digitale Benachrichtigung an den niedergelassenen Arzt bei der Entlassung des Patienten bereits ein Mehrwert im Informationsfluss erzielt werden. Andererseits erlaubt eine elektronische Patientenakte, die von den Leistungserbringern gemeinsam gepflegt wird, eine durchgängige Versorgung des Patienten.

Das in der vorliegenden Arbeit vorgeschlagene Konzept basiert auf der von *Wilson et al.* (2001, 107) genannten zweiten Alternative, unterscheidet sich aber auch von dieser: In dem erarbeiteten Ansatz verwendet jeder Leistungserbringer sein bestehendes Informationssystem weiter, die Informationsaggregation der Patientendaten erfolgt auf der Basis einer virtuellen Patientenakte. Damit wird gewährleistet, dass zwischen den Leistungserbringern keine unvollständigen Informationen auftreten und die Daten über den Patienten dem niedergelassenen Arzt zur Verfügung stehen.

Die in diesem Abschnitt beschriebenen Studienergebnisse lassen den Schluss zu, dass die Kommunikation von Informationen über Institutionsgrenzen hinweg Verbesserungspotenzial besitzt, um eine durchgängige und qualitativ hochwertige Versorgung des Patienten zu erreichen. Dieser Informationsfluss setzt eine Integration auf unterschiedlichen Ebenen voraus, welche im folgenden Abschnitt 4.9.4 beschrieben werden.

4.9.4 Integration

Eine durchgängige Bereitstellung von Informationen für einen Behandlungsprozess kann bei der Entscheidungsfindung für weitere Maßnahmen einen Mehrwert schaffen (Schneider 2001,

N198). Als Grundlage für eine solche Informationsbereitstellung ist auf den im Folgenden dargestellten Ebenen eine geeignete Integration zu schaffen (siehe dazu insbesondere die Ausführungen bei Lenz et al. 2005, 112-117, auf denen die Darstellungen in diesem Abschnitt basieren).

4.9.4.1 Integration auf Datenstrukturebene

Der Wissensumfang in der medizinischen Domäne entwickelt sich ständig weiter (siehe auch Nadkarni et al. 1999, 480). Deshalb ist es erforderlich, bereits etablierte Standards für den Datenaustausch und auch Informationssysteme, die über definierte Schnittstellen Daten importieren oder exportieren, stets geeignet weiterzuentwickeln (siehe auch Nadkarni et al. 1999, 479-480).

Oftmals werden Datenformate von Informationssystemen in Datenbankschemata fest hinterlegt (Lenz et al. 2005, 117). Nachteil dieses Ansatzes ist die kontinuierlich erforderliche Anpassung von Datenbanktabellen, den Anfragen an die Datenbank und der Benutzerschnittstelle (Nadkarni et al. 1999, 480). Um die Datenbankschemata flexibel zu gestalten und damit auf Veränderungen in der Datenmodellierung geeignet reagieren zu können, wird bei *Nadkarni et al.* (1999) ein Ansatz vorgeschlagen, der diese Anforderungen erfüllt. Dieses Konzept wird im Folgenden beschrieben (Nadkarni et al. 1999, 479):

Insbesondere wenn es darum geht, komplexes und ständig in Veränderung befindliches Domänenwissen in Datenstrukturen abzubilden, stellt der EAV-Ansatz (Entity, Attribute, Value) geeignete Maßnahmen zur Verfügung. Die Eigenschaften und Unterschiede dieses Ansatzes im Vergleich zu einer herkömmlichen relationalen Datenmodellierung werden im Folgenden beschrieben: In einer konventionellen Datenbank sind die Daten über mehrere Tabellen verteilt. Diese Tabellen enthalten für jedes Attribut eine separate Spalte. Somit fallen die physikalische und logische Strukturdarstellung der Datenbank nahezu zusammen. Eine Weiterentwicklung in der Modellierung von Datenstrukturen ist das EAV-Konzept, bei dem eine Datenbanktabelle lediglich aus drei Spalten besteht. In diesen Spalten sind die Identifikatoren der Entitäten, Attribute oder Zeiger auf eine Tabelle mit der Attributbeschreibung und der Wert des Attributes gespeichert. Damit fallen die physikalische und die logische Sicht auf die Daten weit auseinander. Das logische Schema einer solchen EAV-Datenbank wird durch Metadaten beschrieben. Die Vorteile dieser Lösung können wie folgt zusammengefasst werden:

- *Flexibilität*: Die Anzahl der Attribute kann sich flexibel weiterentwickeln, ohne das ursprüngliche Datenschema anpassen zu müssen.
- *Geringer Speicheraufwand*: Für Attribute ohne Werte wird kein Speicher reserviert.
- Einfaches physikalisches *Datenformat*
- *Abfragen* sind einfacher zu implementieren als bei konventionellen Strukturen in relationalen Datenbanken.

Datenbankmodelle mit komplexen Objekten und Beziehungen wie in der Objektorientierung, die durch konventionelle Datenbankschemata dargestellt werden, sind häufig im Einsatz (Nadkarni et al. 1999, 480). Um die Vorteile von EAV mit den Anforderungen dieser Daten-

bankmodelle zu kombinieren, wird von *Nadkarni et al.* eine Erweiterung des EAV-Schemas vorgeschlagen, welche im Folgenden erläutert wird (1999, 481): Mit der Erweiterung zum EAV/CR wird das EAV-Konzept mit Eigenschaften aus der Objektorientierung ergänzt. Konkret werden Klassen (Classes, C) und Beziehungen (Relationships, R) hinzugefügt. Das EAV/CR-Konzept umfasst die Einheiten Metadatentabellen, Datentabellen und eine Bibliothek für die Datenmanipulation sowie Benutzerschnittstellen. Mit dem EAV/CR-Konzept wird insgesamt eine komplexe logische Datensicht durch eine einfache physikalische Struktur simuliert (Nadkarni et al. 1999, 491). Damit Konzept wird weiterhin eine Unabhängigkeit des logischen Datenschemas erzielt, lediglich Metadaten werden entsprechend den Entwicklungen in der Domäne angepasst. Insbesondere sind damit elektronische Patientenakten in Zusammenhang mit der geforderten Flexibilität potenzielle Kandidaten für die Handhabung durch das EAV/CR-Konzept (Nadkarni et al. 1999, 480).

Die beschriebenen Arbeiten zeigen auf, dass die Integration auf Datenstrukturebene technisch realisierbar ist. Insbesondere können dabei anpassbare Datenstrukturen vorteilhaft unterstützt werden. Dafür ist jedoch ein zusätzlicher Aufwand erforderlich, um die für typische Datenbanksysteme bereits verfügbare Funktionalität z.B. zur Gewährleistung der Datenintegrität (Lenz et al. 2002, 583) zu implementieren.

4.9.4.2 Integration auf Syntaxebene

Damit Daten von einem Informationssystem in ein anderes übertragen werden können, müssen sie gewissen Regeln entsprechen, um auf syntaktischer Ebene korrekt interpretiert zu werden. Damit diese Voraussetzung erfüllt werden kann, ist entweder ein einheitlicher Standard zu verwenden oder es sind Transformationsregeln anzugeben, die die Daten vom Quell- in ein Zielformat überführen. Der offene XML-Standard bietet hierfür mit seiner Transformationssprache XSLT (XSL Transformations) eine wesentliche Möglichkeit (Lenz et al. 2005, 114). Das Potenzial der XML-Basierung lässt sich auch an den Bemühungen erkennen, ältere Standards wie HL7 Version 2 (siehe Abschnitt 4.11.2.1.1) in das XML-Format zu transformieren (Lenz et al. 2005, 114), so z.B. der in Abschnitt 4.11.1.1 beschriebene VCS-Standard. Da auf etablierte syntaktische Standards in Abschnitt 4.11 detailliert eingegangen wird, wird auf detaillierte Ausführungen in diesem Abschnitt verzichtet.

Insgesamt kann festgestellt werden, dass die Transformation von Standards, insbesondere wenn sie auf XML basieren, aus technischer Hinsicht gegeben ist. Problematisch bleibt in diesem Zusammenhang aber die unterschiedliche semantische Belegung von syntaktischen Elementen.

4.9.4.3 Integration auf Konzeptebene

Die Kompatibilität auf Konzeptebene meint die „semantische Kompatibilität auf Typebene" (Lenz et al. 2005, 114), womit der gewünschte Transfer von in Datenbanken definierten Datentypen und ihren Schemata zwischen unterschiedlichen Anwendungssystemen verstanden wird (Lenz et al. 2005, 114). Obwohl eine Reihe unterschiedlicher Ansätze zum Schema-Matching[24] existiert (siehe Rahm/Bernstein 2001, 344-348), erfordern doch alle manuelle Intervention, wenngleich Werkzeuge zur Unterstützung durch eine grafische Benutzeroberfläche existieren (Rahm/Bernstein 2001, 334). Da Matchings zwischen zwei Schemata nicht

[24] Unter dem Schema-Matching versteht man eine Operation, welche zwei Datenbankschemata als Eingabe erhält und als Ausgabe ein Mapping liefert, welches die beiden Eingaben semantisch in Relation setzt (Rahm/Bernstein 2001, 336).

vollständig automatisch erfolgen können, wird der Fokus bei der Entwicklung von Matching-Algorithmen darauf gelegt, potenzielle Kandidaten zu bestimmen, welche durch Benutzerinteraktion bewertet werden können (Rahm/Bernstein 2001, 337). Die generierten Vorschläge kann der Benutzer akzeptieren, zurückweisen oder anpassen (Rahm/Bernstein 2001, 343), um zum gewünschten Ergebnis zu kommen. Um semantisch hochwertigere Ergebnisse zu erzielen, können unterschiedliche Matching-Algorithmen miteinander verknüpft werden, wobei auch hier eine manuelle Interaktion des Benutzers erforderlich ist (Rahm/Bernstein 2001, 343).

Die technische Realisierung der Konzeptintegration ist mit Algorithmen zum Schema-Matching prinzipiell gegeben. Jedoch ist eine vollständige Automatisierung einer Schematransformation nicht zu erwarten, weshalb die Interaktion des Benutzers ein wesentlicher Bestandteil bleiben wird.

4.9.4.4 Integration auf Terminologieebene

Lenz et al. (2005, 114) definieren die Integration auf Terminologieebene als „semantische Kompatibilität auf Instanzebene". Obwohl bereits einige etablierte Standards zur medizinischen Terminologie existieren (für einen Überblick über ausgewählte Standards siehe Abschnitt 4.11.3), konnte bisher keine Einigung auf einen einheitlichen Standard erzielt werden (Lenz et al. 2005, 115). Für Deutschland sind insbesondere noch keine umfassenden Standards zur Codierung von diagnostischen Maßnahmen und Laborwerten verfügbar (Lenz et al. 2005, 115). Darüber hinaus beschränken sich bisherige Standardisierungsbemühungen auf die medizinischen Bereiche, obwohl auch für weitere Termini z.B. für organisatorische Begriffe Standardisierungen erforderlich sind (Lenz et al. 2005, 115). Insbesondere letzteres Defizit wird bei der Ontologie *OntHoS* (siehe Abschnitt 6.2.5) berücksichtigt, die in der vorliegenden Arbeit zur Implementierung eines agentenbasierten Systems eingesetzt wird. In dieser Ontologie sind neben medizinischen Begrifflichkeiten auch Definitionen aus anderen Bereichen wie z.B. zur Terminvereinbarung eingeschlossen.

Im medizinischen Bereich existiert zwar eine umfassende Menge an semantischen Standards. Jedoch ist bis dato noch keine einheitliche und vollständige Terminologie für den deutschen Raum gegeben. Deshalb sind Transformationen zwischen unterschiedlichen Terminologien bzw. Erweiterungen der vorhandenen Terminologien erforderlich, um eine Integration auf Terminologieebene zu ermöglichen.

4.9.4.5 Kombination von syntaktischer und semantischer Integration

Die Existenz einer Vielzahl unterschiedlicher syntaktischer und semantischer Standards für den Datenaustausch (siehe Abschnitt 4.11.2) bzw. zur Terminologie (siehe Abschnitt 4.11.3) lässt vermuten, dass die Einigung auf einen einheitlichen Standard beschwerlich, wenn nicht sogar unerreichbar ist. Deshalb verfolgen *Pedersen/Hasselbring* (2004) einen Mediator basierten Architekturansatz zur Integration unterschiedlicher Standards. Dieser wird im Folgenden beschrieben:

Mit dem Konzept von *Pedersen/Hasselbring* (2004) wird eine institutionsübergreifende Kommunikation zwischen den jeweiligen Informationssystemen angestrebt. Dazu wird ein Vorschlag unterbreitet, Standards über zu definierende Abbildungsspezifikationen aufeinander abzubilden. Ausgewählte Transformationsvorschriften werden für die Kommunikation und die Dokumentation medizinischer Dokumente angegeben. Um die Semantik der Dokumente zu vereinheitlichen, wird weiterhin ein Terminologiesystem integriert. In einer exemplari-

schen Implementierung ließen sich ausgewählte Abbildungen zwischen unterschiedlichen Kommunikations- und Dokumentationsstandards sowie die Integration eines medizinischen Begriffssystems mit der vorgeschlagenen Architektur realisieren.

Zusammenfassend stellen Pedersen/Hasselbring (Pedersen/Hasselbring 2004) fest, dass für alle Abbildungen der Fallstudie keine vollständigen Transformationsvorschriften konstruiert werden konnten (Pedersen/Hasselbring 2004, 186). Für eine ganzheitliche Transformation ist es erforderlich, die jeweiligen Standards geeignet weiterzuentwickeln (Pedersen/Hasselbring 2004, 186).

Insgesamt kann aus den Ausführungen in den Abschnitten 4.9.4.1 mit 4.9.4.4 geschlossen werden, dass die Integration auf den Ebenen Datenstrukturen und Syntax erfolgreich durchgeführt werden kann. Die automatische Integration auf Konzeptebene ist eingeschränkt, weil für das Schema-Matching manuelle Intervention erforderlich ist. Für eine Integration auf Terminologieebene sind Ergänzungen bzw. eine Einigung auf einen gemeinsamen Standard erforderlich.

Für eine Informationsintegration über die Grenzen von Institutionen hinweg erscheint es sinnvoll, geeignete Transformationen zwischen den Standards zu erarbeiten. Dazu sind die jeweiligen Standards noch adäquat weiterzuentwickeln. Zusammenfassend lässt sich nun feststellen, dass die Informationsintegration im Gesundheitswesen aus technischer Perspektive durchaus realisierbar ist.

4.9.4.6 Integrationsstufen

Lenz et al. (2005, 115) führen vier Stufen der Integration von heterogenen Systemen ein. Grundlage dafür bilden die in den Abschnitten 4.9.4.1 mit 4.9.4.4 beschriebenen Integrationsebenen. Mit diesem mehrstufigen Ansatz wird ermöglicht, die Integration von Informationssystemen in mehreren Schritten vorzunehmen, um einerseits die Komplexität zu reduzieren und andererseits die noch fehlenden Standards berücksichtigen zu können (Lenz et al. 2005, 115). Die Eigenschaften für die jeweiligen Integrationsstufen werden wie folgt beschrieben (Lenz et al. 2005, 115):

- „Verfügbarkeit von Daten aus unterschiedlichen Quellsystemen“ (Lenz et al. 2005, 115)
- „Präsentation patientenbezogener Information in einer übergeordneten Patientenakte“ (Lenz et al. 2005, 115)
- „automatische Übernahme und Weiterverarbeitung strukturierter medizinischer Daten“ (Lenz et al. 2005, 115)
- „Bezugnahme auf patientenbezogene Information im Rahmen von Regeln und Pfaden zur Generierung von Erinnerungshinweisen“ (Lenz et al. 2005, 115)

Tabelle 4.9-1 setzt diese Eigenschaften mit den in den Abschnitten 4.9.4.1 mit 4.9.4.4 dargestellten Integrationsebenen in Relation, um zu beschreiben, ob die jeweiligen Voraussetzungen zur Realisierung der genannten Eigenschaften zu erfüllen sind oder nicht. *Lenz et al.* charakterisieren die in Tabelle 4.9-1 dargestellten Stufen wie folgt (2005, 116):

Die Verfügbarkeit von Daten (Eigenschaft 1 in Tabelle 4.9-1) setzt voraus, dass eine entsprechende Infrastruktur existiert, auf deren Basis die Daten zwischen den verteilten Informationssystemen ausgetauscht werden können. Zusätzlich ist hier anzumerken, dass dabei insbesondere heterogene Hardware-Plattformen zu berücksichtigen sind. Weiterhin ist die Zusammenführung bzw. Transformation unterschiedlicher syntaktischer Standards erforderlich. Eigenschaft (2) wird erreicht, wenn die Daten des Patienten in einer Patientenakte zusammengeführt werden können. Hierfür wird ein geeigneter Kontext benötigt, auf dessen Basis die

Daten des jeweiligen Patienten z.B. über eine globale Patientenidentifikation identifiziert und aggregiert werden können. Um medizinische Daten automatisch weiterverarbeiten zu können, ist für Eigenschaft (3) die konzeptuelle Kompatibilität durchzusetzen. Sollen die Daten für die Eigenschaft (4) auf Anwendungsebene weiterverarbeitet werden, ist eine entsprechende terminologische Basis Voraussetzung.

Voraussetzung / Eigenschaft	Überbrückung verschiedener Plattformen	Kompatibilität der Kontextdaten	Konzeptuelle Kompatibilität	Terminologische Kompatibilität
(1) Verfügbarkeit von Daten	⊕	⊖	⊖	⊖
(2) Präsentation patientenbezogener Information	⊕	⊕	⊖	⊖
(3) Automatische Verarbeitung medizinischer Daten	⊕	⊕	⊕	⊕ ⊖
(4) Bezugnahme auf patientenbezogene Information	⊕	⊕	⊕	⊕

Tabelle 4.9-1: ***Relation zwischen den Eigenschaften der Integration und den jeweiligen Integrationsvoraussetzungen***
Quelle: *Lenz et al.* (2005, 115)

Für eine umfassende Integrationslösung ist die Erfüllung der vier Eigenschaften vollständig vorauszusetzen. Möglichkeiten der konzeptuellen Integration sowie der syntaktischen Kompatibilität sind aus technischer Perspektive bereits gegeben. Die Überbrückung unterschiedlicher Plattformen sowie die Verfügbarkeit einer global eindeutigen Patientenidentifikation werden durch die Einführung der TI für die eGK gegeben sein. Dabei werden zudem plattformunabhängige Konzepte auf der Basis einer Service orientierten Architektur (siehe Abschnitt 4.10.4.1) eingesetzt, womit die Plattformunabhängigkeit gewährleistet ist. Diese kann zusätzlich durch eine geeignete, plattformunabhängige Laufzeitumgebung ergänzt werden. Somit sind Arbeiten einerseits zur Zusammenführung der bisherigen Konzepte zur Integration auf den jeweiligen Ebenen und andererseits zur Gewährleistung der terminologischen Kompatibilität zu fokussieren, um die Integration im Gesundheitswesen voranzutreiben.

Die beschriebenen Integrationsstufen sind Voraussetzung für eine Einführung einer durchgängigen Prozessunterstützung über mehrere Leistungserbringer hinweg (Lenz et al. 2005, 116) und damit für die integrierte Versorgung. Weiterhin können auf dieser Basis zusätzliche Anwendungen zur Verbesserung des informationslogistischen Prinzips, zur Steigerung der Effektivität und Effizienz sowie zur Steuerung des Gesundheitswesens etabliert werden (Lenz et al. 2005, 116). Die grundlegende Voraussetzung für die Integration, die Bereitstellung einer geeigneten Kommunikationsstruktur, wird mit der TI für die eGK im folgenden Abschnitt 4.10 beschrieben.

4.10 Telematikinfrastruktur für die elektronische Gesundheitskarte

4.10.1 Gesetzliche Rahmenbedingungen

In den folgenden Abschnitten 4.10.1.1 mit 4.10.1.3 werden in Anlehnung an die Darstellungen bei *Broy* (2005, 18-21) wesentliche rechtliche Regelungen beschrieben, die für die Einführung eGK bzw. die Rahmenbedingungen für den Einsatz von IT zur Effizienzsteigerung relevant sind (Broy 2005, 20):

Im GKV-Modernisierungsgesetz (Gesetz zur Modernisierung der gesetzlichen Krankenversicherung, Bundesgesetzblatt im Internet 2003) werden die Änderungen im Sozialgesetzbuch (SGB) genannt, die für die Einführung der eGK erforderlich sind. Da in den Anwendungen der eGK auch personenbezogene Daten erhoben, verarbeitet und gespeichert werden, erlangt auch das Bundsdatenschutzgesetz (BDSG) in seiner Anwendung Bedeutung. Die Einführung der Abrechnung in Krankenhäusern durch Fallpauschalen wird durch das Gesetz zur Einführung des diagnoseorientierten Fallpauschalensystems für Krankenhäuser (Fallpauschalengesetz, FPG) geregelt. Mit dieser Regelung kommen Krankenhäuser in den Zugzwang, ihre Prozesse effizienter zu gestalten. Die Einführung der eGK bietet dafür einen geeigneten Ansatzpunkt, vor ihrer eigentlichen Einführung mögliche Potenziale auszuschöpfen, um nach ihrer Einführung von einer adäquaten IT-Infrastruktur und zugehörigen Prozessen profitieren zu können.

Diejenigen Ausschnitte aus den genannten Gesetzen, die für die Einführung der eGK relevant sind, werden in den folgenden Abschnitten 4.10.1.1 mit 4.10.1.3 beschrieben.

4.10.1.1 GKV-Modernisierungsgesetz

Im GKV-Modernisierungsgesetz werden Änderungen zur Modernisierung des Gesundheitswesens gesetzlich beschrieben, welche in weiteren Gesetzen ihre Manifestation erfahren. Demnach wurden Änderungen im SGB V § 291 (Bundesministerium der Justiz 1988) vorgenommen sowie SGB V § 291a (Bundesministerium der Justiz 1988) zusätzlich eingeführt. Die eGK und die mit ihr verbundenen Anwendungsfälle und Prozesse werden durch diese beiden Paragraphen wie folgt spezifiziert:

- Die eGK ist mit einem Lichtbild des Versicherten ausgestattet.
- „Sie muss technisch geeignet sein, Authentifizierung, Verschlüsselung und elektronische Signatur zu ermöglichen“ (SGB V § 291 Abs. 2a, Bundesministerium der Justiz 1988).
- Das Ziel der eGK ist die „Verbesserung von Wirtschaftlichkeit, Qualität und Transparenz der Behandlung“ (SGB V § 291a Abs. 1, Bundesministerium der Justiz 1988).
- „§ 6c des Bundesdatenschutzgesetzes findet Anwendung“ (SGB V § 291a Abs. 2, Bundesministerium der Justiz 1988).
- Der Pflichtteil der eGK umfasst die Angaben zur „Übermittlung ärztlicher Verordnungen in elektronischer und maschinell verwertbarer Form“ (SGB V § 291a Abs. 2, Bundesministerium der Justiz 1988) sowie für den „Berechtigungsnachweis zur Inan-

spruchnahme von Leistungen“ (SGB V § 291a Abs. 2, Bundesministerium der Justiz 1988).

- Darüber hinaus sind für den freiwilligen Teil der eGK „folgende Anwendungen zu unterstützen, insbesondere das Erheben, Verarbeiten und Nutzen von

  1. medizinischen Daten, soweit sie für die Notfallversorgung erforderlich sind,
  2. Befunden, Diagnosen, Therapieempfehlungen sowie Behandlungsberichten in elektronischer und maschinell verwertbarer Form für eine einrichtungsübergreifende, fallbezogene Kooperation (elektronischer Arztbrief),
  3. Daten zur Prüfung der Arzneimitteltherapiesicherheit,
  4. Daten über Befunde, Diagnosen, Therapiemaßnahmen, Behandlungsberichte sowie Impfungen für eine fall- und einrichtungsübergreifende Dokumentation über den Patienten (elektronische Patientenakte),
  5. durch von Versicherten selbst oder für sie zur Verfügung gestellte Daten sowie
  6. Daten über in Anspruch genommene Leistungen und deren vorläufige Kosten für die Versicherten“ (SGB V § 291a Abs. 3, Bundesministerium der Justiz 1988).

- Auf die Daten aus den Pflichtanwendungen dürfen nur folgende Personengruppen zugreifen (SGB V § 291a Abs. 4, Bundesministerium der Justiz 1988):

  1. Ärzte
  2. Zahnärzte
  3. Apotheker, Apothekerassistenten, Pharmazieingenieure, Apothekenassistenten
  4. Berufsmäßige Gehilfen der genannten Personen oder Personen in der Vorbereitung auf einen Beruf der genannten Gruppen
  5. Weitere Erbringer von ärztlich verordneten Leistungen

- Die Daten aus den freiwilligen Anwendungen dürfen nur von den folgenden Personengruppen eingesehen werden (SGB V § 291a Abs. 4, Bundesministerium der Justiz 1988):

  1. Ärzte
  2. Zahnärzte
  3. Apotheker, Apothekerassistenten, Pharmazieingenieure, Apothekenassistenten
  4. Berufsmäßige Gehilfen der genannten Personen oder Personen in der Vorbereitung auf einen Beruf der genannten Gruppen
  5. In Notfällen auch Angehörige eines anderen Heilberufs, der für die Berufsausübung oder die Führung der Berufsbezeichnung eine staatlich geregelte Ausbildung erfordert.
  6. Psychotherapeuten

- „Erheben, Verarbeiten und Nutzen von Daten mittels der elektronischen Gesundheitskarte“ (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988) ist in den freiwilligen Anwendungen nur mit dem Einverständnis des Versicherten erlaubt. Dazu ist, außer in Notfällen, durch „technische Vorkehrungen zu gewährleisten, dass […] der Zugriff

nur durch Autorisierung der Versicherten möglich ist“ (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988).

- Der Zugriff auf in der eGK gespeicherte Daten darf nur in Verbindung mit einem Heilberufsausweis erfolgen (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988).
- Der Benutzer erhält die Möglichkeit, die auf der Karte gespeicherten Daten einzusehen (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988).
- Zum Zwecke der Datenschutzkontrolle sind Zugriffe auf die Daten nach SGB V § 291a Abs. 2 und Abs. 3 im Umfang der letzten 50 Zugriffe zu protokollieren (SGB V § 291a Abs. 6, Bundesministerium der Justiz 1988).
- Nennung der Organisationen, die mit dem Aufbau der TI betraut sind (SGB V § 291a Abs. 7, Bundesministerium der Justiz 1988)
- Details zur Finanzierung der entstehenden Kosten (SGB V § 291a Abs. 7, 7a - 7e, Bundesministerium der Justiz 1988)

4.10.1.2 Bundesdatenschutzgesetz

Da in der TI und insbesondere auf der eGK personenbezogene Daten verarbeitet und gespeichert werden, findet das Bundesdatenschutzgesetz (BDSG, Bundesministerium der Justiz 1990) Anwendung. BDSG § 4 (Bundesministerium der Justiz 1990) regelt dazu folgende Details:

> **„Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten sind nur zulässig, soweit dieses Gesetz oder eine andere Rechtsvorschrift dies erlaubt oder anordnet oder der Betroffene eingewilligt hat“.**

Demnach ist bei der Erhebung, Verarbeitung und Nutzung der personenbezogenen Daten in der TI oder auf der eGK der Zugriff durch den Träger dieses Rechts zu autorisieren.

SGB V § 291a Abs. 2 nennt weiterhin ausdrücklich die Anwendung von BDSG § 6c. Dieser Paragraph liest sich wie folgt (Bundesministerium der Justiz 1990):

(1) „Die Stelle, die ein mobiles personenbezogenes Speicher- und Verarbeitungsmedium ausgibt oder ein Verfahren zur automatisierten Verarbeitung personenbezogener Daten, das ganz oder teilweise auf einem solchen Medium abläuft, auf das Medium aufbringt, ändert oder hierzu bereithält, muss den Betroffenen

   1. über ihre Identität und Anschrift,
   2. in allgemein verständlicher Form über die Funktionsweise des Mediums einschließlich der Art der zu verarbeitenden personenbezogenen Daten,
   3. darüber, wie er seine Rechte nach den §§ 19, 20, 34 und 35 ausüben kann, und
   4. über die bei Verlust oder Zerstörung des Mediums zu treffenden Maßnahmen unterrichten, soweit der Betroffene nicht bereits Kenntnis erlangt hat.

(2) Die nach Absatz 1 verpflichtete Stelle hat dafür Sorge zu tragen, dass die zur Wahrnehmung des Auskunftsrechts erforderlichen Geräte oder Einrichtungen in angemessenem Umfang zum unentgeltlichen Gebrauch zur Verfügung stehen.

(3) Kommunikationsvorgänge, die auf dem Medium eine Datenverarbeitung auslösen, müssen für den Betroffenen eindeutig erkennbar sein."

Demnach sind die Ausgabestellen der eGKn dazu verpflichtet, über deren Funktionsweise zu informieren. Um die Daten auf der eGK einsehen zu können, sind geeignete Terminals für den Patienten zur Verfügung zu stellen.

4.10.1.3 Fallpauschalengesetz

Im Gesetz zur Einführung des Diagnose orientierten Fallpauschalensystems für Krankenhäuser (Fallpauschalengesetz) werden Krankenhäuser zur Abrechnung von erbrachten Leistungen über pauschalisierte Sätze verpflichtet. Ausgehend von einer Einordnung der Krankheit des Patienten wird ein festgelegter Betrag vergütet. Die Vergütung ist demnach nicht mehr von der Dauer des Krankenhausaufenthaltes abhängig, sondern ist allein durch die Klassifizierung der Krankheiten gegeben. Krankenhäuser sollen mit dieser Regelung dazu angehalten werden, die Effizienz der Behandlungsprozesse zu erhöhen. Um jedoch nicht die Behandlungsqualität zu gefährden, werden Krankenhäuser dazu verpflichtet, regelmäßige Qualitätsberichte zu erstellen (Änderung SGB 5 § 137, Broy 2005, 20). Um dennoch die Finanzierungssituation von Krankenhäusern nicht zu gefährden, sind geeignete Maßnahmen zur Steigerung der Effizienz einzurichten, die bspw. durch den gezielten Einsatz von IT realisiert werden können.

4.10.2 Chronologische Darstellung der Entwicklung

Die in Abschnitt 4.10.1 beschriebenen gesetzlichen Rahmenbedingungen sind die Grundlage für die Einrichtung einer TI. Der Aufbau dieser Infrastruktur als Grundvoraussetzung für die Bereitstellung der nötigen Kommunikationsmechanismen für die Anwendungen der eGK erfolgt in mehreren Schritten (siehe dazu auch Broy 2005, 21-22):

- Mit der *Telematikrahmenarchitektur* wird die gesamte Architektur der TI beschrieben. Ziel ist die Schaffung einer Schablone, in die die nächste Stufe der Beschreibung, die Lösungsarchitektur hineinentwickelt wird. Die vom Bundesministerium für Gesundheit und Soziale Sicherung beauftragte Projektgruppe bIT4health (better IT for better health) unter der Führung von IBM Deutschland GmbH legte die erste Fassung der Spezifikation in der Version 1.0 (Projektgruppe bIT4health 2004b) am 22.03.2004 dem Auftraggeber vor. Nach der darauf folgenden Kommentierungsphase wurde eine überarbeitete Version 1.1 (Projektgruppe bIT4health 2004c) am 12.08.2004 in einer bis dato aktuellen Fassung vorgelegt.
- Die *Solution Outline*, die Basis für die folgende Lösungsarchitektur, wurde in der ersten Version 1.0 (Projektgruppe bIT4health 2004a) am 09.07.2004 dem Bundesministerium für Gesundheit und Soziale Sicherung vorgelegt. Die aktuelle Fassung der Solution Outline vom 02.12.2004 liegt derzeit in Version 1.1 (Projektgruppe bIT4health 2004d) vor. Inhalte sind hier „Aussagen über den zeitlichen Rahmen, die Aufwände und die Risiken, die für das Projekt relevant sind" (Broy 2005, 22).
- Die *Lösungsarchitektur* als Grundlage für die Implementierung des eigentlichen Systems wurde von einer Gruppe aus den Fraunhofer Instituten für Software- und Systemtechnik, Arbeitswirtschaft und Organisation sowie Sichere Informationstechnologie mit der Unterstützung der TU Wien (Spezifikation des Konnektors sowie des Kartenterminals) und der Industrie im Auftrag des Bundesministeriums für Gesundheit und Soziale

Sicherung und der Selbstverwaltung erstellt (Broy 2005, 22). Die erste Version 1.0 (Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005) wurde dem Auftraggeber am 14.03.2005 vorgelegt.

- Die gematik (Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH) wurde am 11.01.2005 von den Spitzenorganisationen des deutschen Gesundheitswesens im Auftrag des Bundesministeriums für Gesundheit und Soziale Sicherung gegründet (Broy 2005, 22). Ihre Aufgabe besteht in der Einführung, Pflege und Weiterentwicklung der eGK und der zugrunde liegenden Infrastruktur (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH). Dazu sind alle Komponenten, Dienste, Schnittstellen und Prozesse zu spezifizieren sowie Zertifikate für die Konstruktion der Komponenten an die Industrie zu vergeben (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH). Mit dem Vorliegen der Lösungsarchitektur ist die gematik für den weiteren Prozess zur Einführung und den Betrieb der TI verantwortlich (Broy 2005, 22).

Die in diesem Abschnitt genannten Begriffe der Telematikrahmenarchitektur sowie Lösungsarchitektur werden in den Abschnitten 4.10.3 und 4.10.4 beschrieben.

4.10.3 Telematikrahmenarchitektur

Die Anforderungen für die Entwicklung der TI werden aus der Gesetzgebung, aktuellen Standards sowie aus den Anforderungen der Vertragspartner abgeleitet (Projektgruppe bIT4health 2004b, 15) und fließen in die Gestaltung der generischen Komponenten der Rahmenarchitektur ein. Diese Architektur wird transformiert, um die konkreten Komponenten der Lösungsarchitektur zu erhalten, welche schließlich auf IT-Systemen allokiert und dort installiert werden (Projektgruppe bIT4health 2004b, 15).

In der Rahmenarchitektur wird somit eine generische Struktur beschrieben, in die die Lösungsarchitektur mit ihrer konkreten Ausprägung hineinentwickelt werden soll. Die Bestandteile der Lösungsarchitektur mit ihren Komponenten, Prozessen und Informationen werden dazu auf abstraktem Niveau beschrieben (Broy 2005, 22). Dabei wird eine Reihe von Zielen angestrebt, die sich wie folgt zusammenfassen lassen (Projektgruppe bIT4health 2004b, 14):

- Die individuellen Lösungen für die TI müssen sich in dem von der Rahmenarchitektur vorgegebenen Rahmen befinden.
- Randbedingungen, die für alle Applikationen im Rahmen der TI gelten, werden beschrieben. Darunter werden z.B. Datenschutz- und Sicherheitsmodelle subsumiert.
- Die Rahmenarchitektur berücksichtigt insbesondere diejenigen Artefakte, die in den rechtlichen Grundlagen verankert sind.
- Durch die Rahmenarchitektur werden Schnittstellen zwischen der TI und den Systemen der angrenzenden Umgebung beschrieben.
- Identifikation von Mechanismen für die Erweiterung der TI um neue Dienste
- Schaffung eines möglichst breiten Spielraums zur Umsetzung derjenigen Anforderungen in der Lösungsarchitektur, die in der Rahmenarchitektur beschrieben sind

In die durch diese Ausführungen bestimmten Rahmenbedingungen wird die in Abschnitt 4.10.4 dargestellte Lösungsarchitektur hineinentwickelt.

4.10.4 Telematik-Lösungsarchitektur

In Abbildung 4.10-1 ist die Grobarchitektur der Lösungsarchitektur dargestellt. Diese wird in Anlehnung an die Darstellungen bei *Broy* (2005, 23-24) im Folgenden beschrieben:

Prinzipiell können zwei unterschiedliche Arten von Akteuren identifiziert werden (Broy 2005, 23): Dienstenutzer verwenden die über die TI angebotenen Leistungen der Diensteanbieter. Damit wird eine grundlegende Voraussetzung für die Realisierung der Lösungsarchitektur als Service orientierte Architektur erfüllt (siehe dazu die Ausführungen in Abschnitt 3.2.4.2).

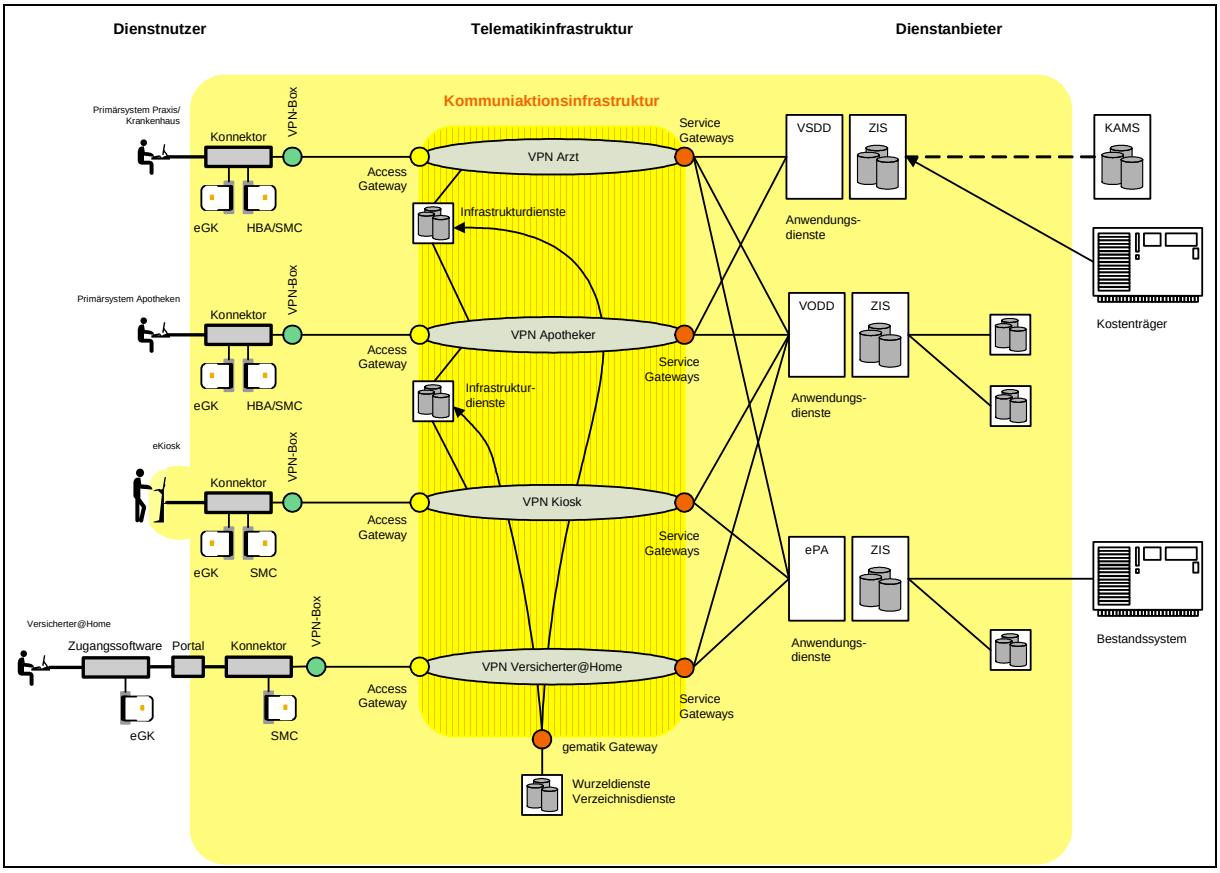

Abbildung 4.10-1: ***Grobarchitektur der Lösungsarchitektur für die TI***
Quelle: In Anlehnung an *Projektgruppe FuE-Projekt „Lösungsarchitektur“* (2005, 29)

Im Folgenden wird die Interaktion der Komponenten für die genannten Akteure beschrieben (siehe für diese Ausführungen auch Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005, 30-32). Die Primärsysteme der Leistungserbringer, z.B. in Arztpraxen und Krankenhäusern, die zur freien Verwendung durch Patienten aufgestellten Kiosksysteme sowie geeignete Zugangs-Software zusammen mit einem Portal für den Zugang zu angebotenen Diensten durch einen PC werden über einen Konnektor an die Kommunikationsinfrastruktur angebunden. Dieser Konnektor kommuniziert mit den genannten Primärsystemen sowie dem Kartenterminal für die eGK, den Heilberufsausweis sowie eventuell vorhandene Secure Module Cards[25]

[25] Für eine Beschreibung der genannten Karten siehe die Ausführungen in den Abschnitten 4.10.4.3 mit 4.10.4.5.

(SMC). Die Kommunikation zwischen dem Konnektor und den Karten-Terminals bzw. der TI erfolgt für den Benutzer transparent (Broy 2005, 25) und wird automatisch verschlüsselt.

An den Konnektor ist eine VPN-Box angebunden. Über ein Access Gateway wird die Verbindung zur Kommunikationsinfrastruktur hergestellt. Access Gateways gewähren nur registrierten Nutzern den gewünschten Zugang zur Kommunikationsinfrastruktur. Ein Zertifikat im verwendeten Zugangsknoten bestimmt die Zuordnung zu einem passenden VPN. Mit der Zuordnung zu einem dedizierten VPN wird eine Benutzerrolle festgelegt.

Im Service Gateway ist eine Zuordnung zwischen vorliegender Rolle und den gewährten Berechtigungen zur Inanspruchnahme von Anwendungsdiensten hinterlegt. Diese Berechtigungen legen fest, welche Dienste aus dem VPN des Benutzers aufgerufen werden dürfen. Weil Access Gateways und Service Gateways über einen Trusted Backbone kommunizieren, sich die Gateways gegenseitig authentifizieren und über VPN miteinander verbunden sind, wird gewährleistet, dass nur diejenigen Anwender, die über Access Gateways die Dienste der Kommunikationsinfrastruktur und die Anwendungsdienste aufrufen, zur Ausführung berechtigt sind (Broy 2005, 24; Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005, 30-31).

Das gematik Gateway bietet den Zugriff auf Wurzel- und Verzeichnisdienste, die für die Administration des Netzes erforderlich sind. Ausgezeichnete VPNs können Infrastrukturdienste aufrufen.

Die Anwendungsdienste wie der Zugang zur elektronischen Patientenakte (ePA), der Verordnungsdatendienst (VODD) oder der Versichertenstammdatendienst (VSDD) können über die Service Gateways in Anspruch genommen werden. Die Anwendungsdienste greifen über eine einheitliche Zugangs- und Integrationsschicht (ZIS) auf die relevanten Daten zu. Mit dieser Schicht wird eine einheitliche Rechteverwaltung zum Zugriff auf die Daten umgesetzt. Damit werden den Benutzern adäquate Rechte zugeordnet. Die ZIS verbirgt außerdem die tatsächliche Verteilung der Daten und erzielt damit die gewünschte Speichertransparenz. Mit dieser Kapselung wird die künftige Erweiterbarkeit zur Anbindung weiterer externer Systeme erleichtert, weil die Schnittstellen der Anwendungsdienste nicht angepasst werden müssen.

4.10.4.1 Telematikinfrastruktur als Service orientierte Architektur

Die TI ist als eine SOA (siehe Abschnitt 3.2.4.2) konzipiert (Neuhaus/Deiters/Wiedeler 2006, 333 und 339; Caumanns et al. 2006, 343-346): Dazu werden die in der TI realisierten Funktionalitäten durch Dienste zur Verfügung gestellt, welche in die Klassen Anwendungsdienste, Infrastrukturdienste, dezentrale Dienste und Sicherheitsdienste (Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005, 43) eingeteilt werden können. Diese werden wie folgt definiert (Projektgruppe FuE-Projekt „Lösungsarchitektur“ 2005, 36-39):

- *Anwendungsdienste* realisieren diejenigen Dienste, durch die Anwendungsdaten gelesen, geändert, geschrieben und gelöscht werden. Es werden darunter die Dienste subsumiert, die die Anwendungen wie die elektronische Patientenakte, den Arztbrief oder die Verwaltung der Patientendaten realisieren.

- *Dezentrale Dienste* sind im Konnektor implementiert und kapseln Anwendungs- und Infrastrukturdienste oder realisieren Schnittstellen für den Zugriff des Konnektors auf Dienste der Server-Seite. Beispiele hierfür sind Karten-Dienste, die Zugriffe auf die Karten realisieren, oder Kartenterminal-Dienste, die den Zugriff auf die Kartenterminals implementieren.
- *Infrastrukturdienste* stellen Basismechanismen – vergleichbar mit denen einer Middleware-Implementierung – bereit und realisieren über PKI-Dienste die Sicherheit und Integrität der Kommunikation.
- Lokale *Sicherheitsdienste* realisieren die Authentifizierung, Autorisierung, Integrität, Nichtabstreitbarkeit, Vertraulichkeit und Zertifikatsvalidierung.

Obwohl auf die dynamische Dienstbindung z.B. über UDDI in der TI verzichtet wird und z.B. SOAP nur optional eingesetzt werden kann, kann dennoch festgestellt werden, dass über die Zusammenwirkung der Dienste ein Enterprise Service Bus gebildet wird (Caumanns 2006, 346). Dazu können folgende Aspekte festgehalten werden (Caumanns 2006, 346):

- „Entkopplung von Diensten“ (Caumanns 2006, 346)
- „automatische Dienstkonfiguration“ (Caumanns 2006, 346)
- „dynamische Anpassung des Service Levels beim Nachrichtenaustausch“ (Caumanns 2006, 346)

Die TI besitzt als Service orientierte Architektur zur Realisierung der Pflicht- und freiwilligen Anwendungen somit auch das Potenzial für die Unterstützung weiterer Mehrwertdienste. Ein Überblick über mögliche Szenarien wird bei Neuhaus/Deiters/Wiedeler (2006) gegeben. Auch das in der vorliegenden Arbeit dargestellte Lösungskonzept kann als ein solcher Mehrwertdienst betrachtet werden.

Mit der Realisierung der TI nach SOA-Prinzipien können die in Abschnitt 3.2.4.2 dargestellten Charakteristika und damit die in Abschnitt 3.4.2.5 ausgeführten Potenziale zur Modularisierung und Unterstützung der losen Kopplung erreicht werden. Obwohl diese Potenziale entsprechend den Ausführungen in Abschnitt 3.4.2.5 beschränkt sind, kann einer SOA zum gegenwärtigen Zeitpunkt dennoch eine adäquate Flexibilität zur Realisierung von Integrationsszenarien attribuiert werden. Somit kann der TI für die eGK insgesamt ein wesentliches Potenzial zur Verbesserung der Versorgungsqualität und zur Effizienzsteigerung im Gesundheitswesen beigemessen werden, weil die technischen Voraussetzungen dafür in einer geeigneten Architektur unterstützt werden.

Elemente der TI, mit denen die Anwender im Versorgungsalltag am häufigsten konfrontiert werden, sind Kartenlesegeräte, die eGK, der HBA sowie die SMC. Diese werden deshalb in den folgenden Abschnitten 4.10.4.2 mit 4.10.4.5 beschrieben.

4.10.4.2 Kartenlesegerät

In diesem Abschnitt werden die Anforderungen an Kartenlesegeräte, die in der TI zum Lesen und Beschreiben der eGK, des HBAes und von SMCs eingesetzt werden, sowie ihre stufenweise Einführung über Szenarien beschrieben.

Um eine Zertifizierung für ein Kartenterminal für den Einsatz in der TI zu erhalten, wird die Erfüllung der folgenden Eigenschaften vorausgesetzt:

- Gewährleistung einer *fehlerfreien Kommunikation* mit den eingelegten Karten sowie Unterstützung der erforderlichen *Sicherheitseigenschaften* (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH: Version 1.0.0 vom 28.04.2006 2006, 5)

- Zur Unterstützung der *Netzwerkfähigkeit* sind in der Ausführung der Terminalgeräte prinzipiell zwei unterschiedliche Lösungen möglich: Netzwerkfähige Geräte werden direkt an ein LAN (Local Area Network) angeschlossen (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH: Version 1.0.0 vom 28.04.2006 2006, 9). Virtuelle Terminals hingegen sind eine Kombination eines nicht netzwerkfähigen Lesegerätes und einer passenden Software-Emulation, welche die Netzwerkfähigkeit des Gerätes abbildet (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH: Version 1.0.0 vom 28.04.2006 2006, 9).
- Für den Einsatz in der TI müssen die Lesegeräte die *SICCT-Spezifikation* (Secure Interoperable ChipCardTerminal, TeleTrusT Deutschland e.V. 2006) erfüllen.

Da die Anwendungen der TI stufenweise etabliert werden, ist ein geeignetes Vorgehen für die Einführung der Kartenterminals vorgesehen. Für den Übergang von der jetzigen Krankenversichertenkarte (KVK) zur eGK werden deshalb die folgenden Szenarien differenziert:

- Multifunktionale Kartenterminals (MKT, ohne SICCT-Spezifikation) sollen sowohl die Daten von den Karten der TI als auch von bisherigen KVKn verarbeiten können.
- In dem Szenario MKT+ werden die Lesegeräte direkt an die Rechner mit Primärsystemen angeschlossen, wobei dabei auf die Funktionalität des Konnektors und somit auf die Anwendungen der TI verzichtet wird. Deshalb kann hier nur die Funktionalität wie bei KVKn unterstützt werden.
- In weiteren Ausbaustufen werden die Funktionalitäten der TI schrittweise eingebunden. Diese Ausbaustufen sind zum Zeitpunkt der Erstellung der vorliegenden Arbeit noch nicht spezifiziert.

Die Spezifikationen der TI sehen für die Nutzung der durch sie zur Verfügung gestellten Funktionalitäten jeweils die Verwendung der Karten eGK, HBA und SMC vor. Damit erlangen Aspekte zur geeigneten Integration dieser Karten in die Arbeitsabläufe der beteiligten Akteure Relevanz. Weil die künftigen Abläufe insbesondere der Leistungserbringer wesentlich durch den Umgang mit den Gesundheitskarten geprägt sind, sind dafür geeignete Mechanismen zu erarbeiten, die eine vereinfachte Integration der Karten in Arbeitsprozesse erlauben. Dazu wird bei Schweiger et al. (2007b) und Mauro et al. (2008) ein Ansatz zur zentralen Verwaltung von Gesundheitskarten vorgeschlagen. Damit werden Nachteile des dezentralen gematik-Ansatzes beseitigt, bei dem an jedem Arbeitsplatz Kartenterminals einzurichten sind und Verluste in der Arbeitseffizienz identifiziert werden können. Darüber hinaus können mit dem zentralen Ansatz Funktionalitäten zur Verfügung gestellt werden, die über den in der TI spezifizierten Funktionsumfang hinausgehen. Somit können die durch die Einführung der TI erforderlichen Veränderungen dazu herangezogen werden, Prozesse im Arbeitsalltag ggf. zu überarbeiten, um Steigerungen in der Arbeitseffizienz und -effektivität zu erreichen.

4.10.4.3 Elektronische Gesundheitskarte

Die elektronische Gesundheitskarte (eGK) wird die bisherige Krankenversichertenkarte (KVK) ersetzen und Daten sowie Funktionalitäten zur Verfügung stellen, die über den Umfang der KVK hinaus gehen. Unterschieden werden dabei zwei differenzierte Anwendungsarten: obligatorische und freiwillige.

Zum verpflichtenden Teil (SGB V § 291a Abs. 2, Bundesministerium der Justiz 1988) gehören die Verwaltungsdaten, die bisher auf der Versichertenkarte gespeichert sind. Für den lesenden und schreibenden Zugriff auf diese Daten wird lediglich die eGK des Patienten vorausgesetzt (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988). Als weitere obligatorische Anwendungen kommen Funktionalitäten zur Unterstützung von ärztlichen Verordnungen sowie ein Aufdruck zum Nachweis der Inanspruchnahme von Versicherungsleistun-

gen im europäischen Ausland hinzu. Beim Lesen und Schreiben von ärztlichen Verordnungen ist neben der eGK auch ein Heilberufsausweis (siehe Abschnitt 4.10.4.4) erforderlich (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988).

Freiwillige Daten auf der Gesundheitskarte können die Versorgungsqualität erhöhen, indem diese als Grundlage für die Realisierung der folgenden Anwendungen fungieren (SGB V § 291a Abs. 3, Bundesministerium der Justiz 1988):

- Medizinische Daten für die Notfallversorgung
- Elektronischer Arztbrief
- Arzneimitteltherapiesicherheit
- Elektronische Patientenakte
- Patientenfach
- Patientenquittung

Für die Anwendung Patientenfach ist für den Zugriff auf die eGK-Daten neben der eGK eine Signaturkarte mit qualifizierter elektronischer Signatur erforderlich (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988). Voraussetzung für den Zugriff auf die Daten der übrigen freiwilligen Anwendungen ist neben der eGK ein Heilberufsausweis (SGB V § 291a Abs. 5, Bundesministerium der Justiz 1988).

Die Karte umfasst in der aktuellen Spezifikation für die eGK einen 59.228 Bytes großen Speicher (gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH 2006, 8), um alle Anwendungen der eGK zu unterstützen. Weil dieser Speicherumfang begrenzt ist, sind insbesondere die umfangreicheren Daten für die freiwilligen Anwendungen auf zentralen Servern abzulegen.

Voraussetzung für die Nutzbarkeit der Anwendungen sowie die Akzeptanz der eGK sind adäquate Verfahren, die Berechtigungen verwalten und überprüfen können (Caumanns 2006, 323). *Caumanns* (2006) beschreibt die zur Realisierung geplanten Mechanismen detailliert, welche über ein „ticketbasiertes, virtuelles Dateisystem“ (Caumanns 2006, 323) umgesetzt werden können. Diese Mechanismen stehen jedoch nicht im Fokus der vorliegenden Arbeit.

Mit der Einführung der eGK wird dem informationellen Selbstbestimmungsrecht des Patienten Rechnung getragen, indem er über die Festlegung, wer welche Daten einsehen darf, in den Behandlungsprozess integriert wird. Über den lesenden Zugriff des Patienten auf Daten ist weiterhin zu erwarten, dass der gesamte Behandlungsprozess verbessert wird. Außerdem kann die Versorgungsqualität durch die freiwilligen Anwendungen gesteigert werden, weil bspw. die bisherige Krankheitsgeschichte über eine geeignete elektronische Patientenakte abrufbar ist. Insofern werden dem Patienten, geeignete Sicherheitsmaßnahmen vorausgesetzt, durch die eGK wesentliche Mehrwerte zur Verfügung gestellt.

4.10.4.4 Heilberufsausweis

Heilberufsausweise (HBA) werden an Leistungserbringer im Gesundheitswesen (für eine detaillierte Übersicht siehe Abschnitt 4.10.1.1) ausgegeben und besitzen eine qualifizierte elektronische Signatur (Caumanns et al. 2006, 343), mit der handschriftliche Unterschriften ersetzt werden, um die Anwendungen der eGK zu unterstützen. Sie sind damit das Gegen-

stück zur eGK, um rechtsverbindliche Aufgaben vornehmen zu können. Ein HBA ist weiterhin für den lesenden und schreibenden Zugriff auf Daten der eGK bei bestimmten Anwendungen erforderlich. Diese Anforderungen sind in Abschnitt 4.10.4.3 beschrieben.

4.10.4.5 Secure Module Card

Secure Module Cards (SMC) werden für Institutionen im Gesundheitswesen sowie für medizinisches Hilfspersonal ausgegeben (Caumanns et al. 2006, 343). Diese entsprechen in ihrer Funktionalität im Wesentlichen einem HBA, besitzen im Gegensatz zu diesem jedoch keine Funktionalität zur qualifizierten elektronischen Signatur (Caumanns et al. 2006, 343). Eine Assoziation mit einer bestimmten Person wie bei einem HBA ist mit einer SMC somit nicht gegeben, vielmehr ist letztere für eine Institution ausgestellt.

4.10.4.6 eKiosk und Versicherter@Home

eKioske zielen auf die Unterstützung des informationellen Selbstbestimmungsrechts (Bernnat/Booz Allen Hamilton GmbH 2006, 81; SGB V § 291a, Bundesministerium der Justiz 1988) des Patienten ab. Zum Zeitpunkt der Erstellung der vorliegenden Arbeit liegen keine Spezifikationen der eKiosk-Funktionalität vor (Bernnat/Booz Allen Hamilton GmbH 2006, 81); deshalb wird für die folgenden Ausführungen in Analogie zu den Einschätzungen bei *Bernnat* (2006, 81) u.a. die Funktionalität der eKiosk-Anwendung zum Lesen der elektronischen Patientenakte betrachtet: Dazu werden neben der eGK ein Konnektor, eine SMC sowie eine SICCT-Komponente (Bernnat/Booz Allen Hamilton GmbH 2006, 82) und zur Realisierung der Benutzeroberfläche ein Browser (Bernnat/Booz Allen Hamilton GmbH 2006, 83) vorausgesetzt.

Auch für die Anwendung Versicherter@Home, bei der der Patient ohne SICCT-Komponente und Konnektor auf Daten zugreifen kann, liegen zum Zeitpunkt der Erstellung der vorliegenden Arbeit keine Spezifikationen vor (Bernnat/Booz Allen Hamilton GmbH 2006, 83). Unter der Annahme, dass kein Konnektor und keine SICCT-Komponente beim Patienten verfügbar sind, kann nur auf unverschlüsselte Daten zugegriffen werden, die sich auf die Stammdaten des Patienten beschränken (Bernnat/Booz Allen Hamilton GmbH 2006, 84). Unter diesen Voraussetzungen stellt die Applikation Versicherter@Home nur einen sehr eingeschränkten Mehrwert für den Patienten dar. Deshalb wird in der vorliegenden Arbeit eine mögliche Realisierung der eKiosk-Funktionalität mit einer Anbindung an das AMD angestrebt.

Mit den Anwendungen eKiosk und Versicherter@Home wird die aktive Integration des Patienten in den Behandlungsprozess angestrebt. Weil beide Applikationen zum gegenwärtigen Zeitpunkt nicht spezifiziert sind, werden bei Bernnat (2006, 82 und 84) angemessene Annahmen getroffen. Diese schränken aber die gewünschte Beteiligung des Patienten ein: eKioske werden zwar den Zugriff z.B. auf die elektronische Patientenakte erlauben, diese Funktionalität ist aber an die Lokalität gebunden, an der sich der eKiosk befindet. Bei dem angenommenen Szenario für die Applikation Versicherter@Home ist fragwürdig, ob die für den Patienten einsehbaren Stammdaten für ihn von Interesse bzw. Nutzen sind.

Für den Aufbau von Telematik-Lösungen im Gesundheitswesen ist die Berücksichtigung geeigneter und etablierter Standards relevant. Im folgenden Abschnitt 4.11 werden deshalb wesentliche Standards beschrieben.

4.11 Standards im Gesundheitswesen

In Anlehnung an die Strukturierung bei *Lenz et al.* (2005, 106-108) werden in den folgenden Abschnitten 4.11.1 mit 4.11.4 Standards für den medizinischen Bereich beschrieben, die Relevanz für die IT basierte Verarbeitung besitzen. Die Strukturierung der Standards erfolgt dabei entsprechend der in Tabelle 4.9-1 beschriebenen Integrationsstufen. Für weitere Ausführungen zu Standards, insbesondere für den syntaktischen und semantischen Bereich, sowie einer Darstellung ihrer Zusammenhänge wird auf *Sunyaev et al.* (2008) verwiesen.

4.11.1 Standards zur Infrastruktur

4.11.1.1 VCS

Der VCS-Standard (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006a; VDAP (Verband Deutscher Arztpraxis-Softwarehersteller e.V.) zielt auf die Integration von Arztinformationssystemen der VDAP-Mitglieder und möchte so einen Beitrag zur Unterstützung eines übergreifenden Behandlungsprozesses liefern. Für eine Integration dieser Systeme wird eine Kompatibilität mit der VCS-Schnittstelle vorausgesetzt (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006c).

VCS ist offen und kann so auch von anderen Akteuren im Gesundheitswesen eingesetzt werden (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006a): VCS spezifiziert einerseits den Übertragungsweg und andererseits den Inhalt sowie die Struktur der übermittelten Dokumente, wobei die Kommunikation über existierende Internet-Standards abgewickelt wird (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006b). Das aktuelle Datenaustauschformat ist BDT (siehe dazu Abschnitt 4.11.2.2), zukünftig wird die Struktur der Inhalte möglicherweise über XML bestimmt (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006b).

Die Interaktion zum Datenaustausch zwischen zwei Leistungserbringern umfasst die folgenden Schritte (VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) 2006b): Vor dem Versand der Dokumente werden diese für den Empfänger verschlüsselt und vom Absender signiert. Der Versand erfolgt als ein Email-Dokument. Beim Abrufen der Dokumente wird die Signatur überprüft und es werden die Dokumente mit dem Schlüssel des Empfängers entschlüsselt. Nach erfolgreicher Überprüfung der Signatur wird eine Empfangsbestätigung an den Absender gesendet.

Mit VCS wird ein offener Standard zum Datenaustausch zwischen Leistungserbringern im ambulanten Sektor definiert. Obwohl damit die Möglichkeit unterstützt wird, Daten entsprechend dem Behandlungsprozess zwischen Leistungserbringern auszutauschen, ist dennoch kritisch anzumerken, dass mit der Verschlüsselung der Email an den jeweiligen Empfänger die freie Arztwahl der Patienten beschränkt wird. Insofern ist eine geeignete Weiterentwicklung von VCS erforderlich.

4.11.1.2 PaDok®

Das Ziel von PaDok[26] (Patientenbegleitende Dokumentation) ist die Bereitstellung „von Daten […] auf einem gemeinsamen Server“ (Fraunhofer Institut für Biomedizinische Technik (Arbeitsgruppe Medizin-Telematik) 2000) und das „Weiterleiten […] dieser Daten an einen autorisierten […] Empfänger“ (Fraunhofer Institut für Biomedizinische Technik (Arbeitsgruppe Medizin-Telematik) 2000). Die so bereitgestellten Daten müssen derart beschaffen sein, dass das Wahlrecht des Patienten für einen präferierten Leistungserbringer nicht eingeschränkt wird. Dazu wird ein spezielles Verfahren mit der Umschlüsselung der Daten auf dem Server für den gewählten Empfänger eingesetzt (Fraunhofer Institut für Biomedizinische Technik (Arbeitsgruppe Medizin-Telematik) 2000).

PaDok beseitigt das Defizit der Einschränkung der freien Arztwahl von VCS, indem solche Verschlüsselungsverfahren eingesetzt werden, die einerseits die erforderlichen Sicherheitseigenschaften erfüllen und andererseits die Einsichtnahme der medizinischen Daten durch jeden beliebigen durch den Patienten autorisierten Leistungserbringer unterstützen können. Auf dieser Basis werden mit D2D, beschrieben im folgenden Abschnitt 4.11.1.3, Applikationen zum Datenaustausch bestimmt.

4.11.1.3 D2D

Die Kommunikationsplattform D2D (Doctor to Doctor, KVNo Kassenärztliche Vereinigung Nordrhein 2005) wurde von der Kassenärztlichen Vereinigung Nordrhein initiiert, um die digitale Kommunikation zwischen Ärzten zu ermöglichen. Das Ziel ist der sichere Datentransfer zwischen den Teilnehmern. Die Grundlagen für diesen Ansatz wurden vom Fraunhofer-Institut für Biomedizinische Technik (St. Ingbert/Saar) mit dem PaDok-Ansatz (siehe Abschnitt 4.11.1.2) bereitgestellt. Auf dieser Basis werden von der D2D-Plattform Anwendungen wie z.B. eArztbrief, eÜberweisung, eKrankenhauseinweisung oder eFallakte (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 3) und damit eine Sektoren übergreifende Kommunikation unterstützt.

Die Architektur der D2D-Lösung ist in Abbildung 4.11-1 dargestellt und wird im Folgenden erläutert: Im Zentrum der Anwendungen steht der D2D-Messaging-Server. Auf diesem werden die Patientendaten in verschlüsselter Form für einen begrenzten Zeitraum zur Verfügung gestellt. Die Verbindung zwischen dem D2D-Client auf dem Client-Rechner z.B. in der Praxis eines niedergelassenen Arztes und dem Messaging-Server erfolgt entweder über eine Direkteinwahl oder eine VPN-Verbindung und basiert auf TCP/IP (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 4). Beim Einstellen von Dokumenten auf dem Server wird zunächst ein Hauptdokument mit den administrativen Informationen erstellt, welche den Patienten, den Arzt und den Vorgang bezeichnen (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 5). An dieses Hauptdokument können weitere Dokumente angefügt werden (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 5). Die D2D-Lösung verwendet zur Übertragung der Dokumente XML bzw. SCIPHOX (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 6-7). Um die freie Arztwahl zu unterstützen, wird ein besonderes Verschlüsselungsverfahren eingesetzt, das gewährleistet, dass die Daten auf dem Server beim Umschlüsseln nicht in den Klartext übergeführt werden (siehe KVNo Kassenärztliche Vereinigung Nordrhein

[26] Weitere Informationen können auf den folgenden Seiten abgerufen werden: http://padok.ibmt.fhg.de, zugegriffen am 14.03.2007.

2005, 4-5). Dabei erhält der Patient einen Teil des öffentlichen Schlüssels, mit dem das Dokument verschlüsselt wurde und der gewählte Arzt in der Lage ist, das nicht adressierte Dokument auf dem Server wieder zu entschlüsseln. Zur Entschlüsselung beim Abrufen der Daten auf dem Server müssen Schlüsselteile eines autorisierten Empfängers und des Patienten zum gleichen Zeitpunkt vorliegen (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 4-5).

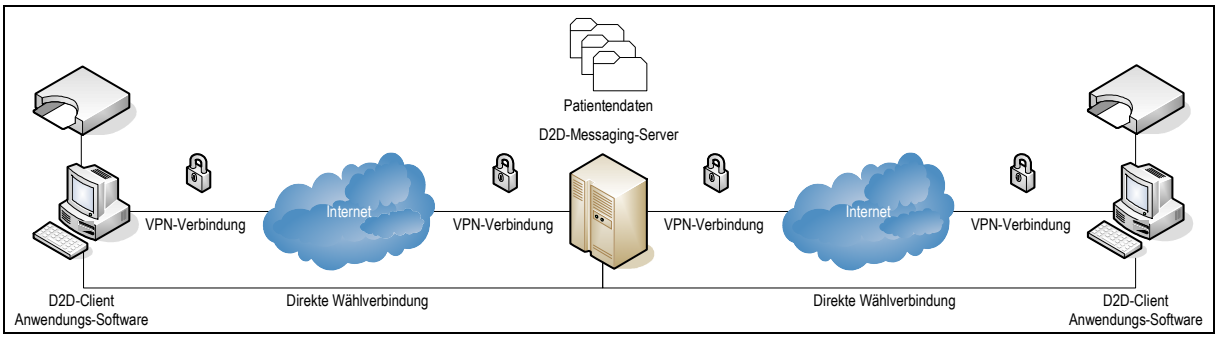

Abbildung 4.11-1: ***D2D-Architektur***
Quelle: Eigene Darstellung, abgeleitet aus den Ausführungen in Abschnitt 4.11.1.3

Zur Ver- und Entschlüsselung sowie zur Erstellung und Überprüfung von Signaturen werden symmetrische und asymmetrische Verfahren eingesetzt. Die asymmetrische Verschlüsselung und Signatur werden durch eine geeignete Smartcard umgesetzt bzw. können durch den kommenden HBA unterstützt werden (KVNo Kassenärztliche Vereinigung Nordrhein 2005, 5 und 8).

Im Wesentlichen wird durch den D2D-Ansatz eine Email-Kommunikationsplattform geschaffen, die den Austausch von Patientendaten zwischen Ärzten unterschiedlicher Sektoren erlaubt. Insbesondere wird dabei das Selbstbestimmungsrecht des Patienten zur Arztwahl unterstützt.

4.11.1.4 Microsoft eHIP

Die Microsoft eHIP-Plattform (eHealth Interoperability Plattform, Microsoft Deutschland GmbH o.J.-a) zielt auf die Unterstützung der integrierten Versorgung ab. Dabei werden proprietäre Produkte auf der Basis von offenen Standards wie HL7 (Microsoft Deutschland GmbH o.J.-b, 3), XML oder Web Services in Kombination mit SOAP (Microsoft Deutschland GmbH o.J.-a, 2-3) eingesetzt, um die Primärsysteme der verschiedenen Leistungserbringer zu verknüpfen. Die interagierenden Komponenten orientieren sich dabei an dem Prinzip der Service orientierten Architektur (Microsoft Deutschland GmbH o.J.-a, 3), indem in der Integrations- und Kommunikationsplattform (Microsoft Deutschland GmbH o.J.-a, 3) geeignete Dienste zur Verfügung gestellt werden. In einer Weiterentwicklung der eHIP-Plattform werden die Integration von HBA und eGK sowie die Entwicklung einer geeigneten Sicherheitsinfrastruktur berücksichtigt (Microsoft Deutschland GmbH o.J.-a, 3-4).

Das PaDok-Konzept beseitigt das Defizit der freien Arztwahl von VCS. In D2D wird ersteres um Applikationen erweitert. In Anbetracht der Ausführungen zur Bewertung von Architekturen in Abschnitt 3.4.2.1 erscheint das bei D2D realisierte Client-Server-Prinzip jedoch zu wenig flexibel, um den Anforderungen der Integration gerecht zu werden. eHIP ist zwar entsprechend einer Service orientierten Architektur konstruiert und ist damit im Vergleich zu D2D in einem Integrationsszenario flexibler, proprietäre Lösungen erschweren aber die gewünschte Integration über die Sektoren hinweg. Außerdem ist auch die durch eine SOA induzierte Flexibilität

eingeschränkt (siehe dazu Abschnitt 3.4.2.5). Die beschriebenen Kommunikationsinfrastrukturen stellen deshalb keine anzustrebende Lösung dar, obwohl geeignete Konzepte zur Entwicklung einer TI wieder verwendet werden können.

4.11.2 Standards zum Datenaustausch

4.11.2.1 Standards im stationären Bereich

4.11.2.1.1 HL7

Die Organisation hinter HL7 (Health Level Seven®, Health Level Seven Inc. Ann Arbor MI 2005) ist eine vom ANSI (American National Standards Institute) akkreditierte Arbeitsgruppe (Health Level Seven Inc. Ann Arbor MI 2005). HL7 besitzt die folgenden, wesentlichen Eigenschaften (Health Level Seven Inc. Ann Arbor MI 2005): Ziel ist der standardisierte Austausch von klinischen und administrativen zwischen verschiedenen Informationssystemen. Dazu wird ein umfassender Standard entwickelt, der nicht nur auf bestimmte Abteilungen beschränkt ist, sondern die Anforderungen in einem gesamten Klinikum berücksichtigt:

Im Rahmen von HL7 wurden bislang unterschiedliche Standards entwickelt bzw. akkreditiert. Im Folgenden werden die wesentlichen Versionen 2 und 3 sowie mit CDA ein XML basiertes Format zur Strukturierung von Dokumenten beschrieben:

Die Familie der HL7-Version 2 ist akkreditiert vom ANSI. Version 2.5 ist die zum Zeitpunkt der Erstellung der vorliegenden Arbeit letzte vom ANSI akkreditierte Version. Das dabei zugrunde liegende Kommunikationsmodell stützt sich auf die Ereignissteuerung, wobei die einzelnen Nachrichten rein sequenzbasiert sind (Heitmann/Schweiger/Dudeck 2003, 196).

HL7 Version 3 befindet sich aktuell in Entwicklung und basiert im Gegensatz zur Version 2 ausschließlich auf XML. Das HL7-Referenzinformationsmodell (RIM) ist ein wesentlicher Bestandteil dieser Version und besitzt die folgenden Eigenschaften (Health Level Seven Inc. Ann Arbor MI 2005):

- Definition einer semantischen und terminologischen Grundlage für die in HL7-Nachrichten beinhalteten Felder
- Beschreibung von klinischen Daten sowie des Lebenszyklus von Ereignissen, die innerhalb einer Nachricht oder einer Menge von verknüpften Nachrichten auftreten
- Spezifikation der Verknüpfungen zwischen den Feldern einer HL7-Nachricht

Obwohl HL7 CDA einen Standard für Dokumente (siehe dazu Abschnitt 4.11.4) darstellt, werden die Eigenschaften wegen des Zusammenhangs mit HL7 dennoch in diesem Abschnitt beschrieben. HL7 CDA definiert die Struktur von klinischen Dokumenten im XML-Format, um z.B. Entlassbriefe zwischen unterschiedlichen Systemen auszutauschen (Health Level Seven Inc. Ann Arbor MI 2005). CDA-Dokumente stützen sich auf das RIM (Heitmann/Schweiger/Dudeck 2003, 197) und das HL7-Vokabular (Health Level Seven Inc. Ann Arbor MI 2005). Damit werden solche Dokumente zu einer semistrukturierten Sammlung (Boeker/Jentzsch/Klar 2004, 1531) von semantisch definierten Sachverhalten. Auf der Basis von XML können CDA-Dokumente sowohl automatisiert weiterverarbeitet als auch für

menschliche Benutzer durch ein entsprechendes Anzeigeprogramm visualisiert werden (Health Level Seven Inc. Ann Arbor MI 2005).

Ein CDA-Dokument (siehe Abbildung 4.11-2) besteht aus den beiden Elementen Header und Body (Heitmann/Schweiger/Dudeck 2003, 197), deren Inhalte in Anlehnung an *Heitmann/Schweiger/Dudeck* (2003, 197-198) wie folgt beschrieben werden: Im Header werden Metadaten zum Dokument wie Versionsinformationen, Daten zum vorliegenden Fall, Informationen zum Leistungserbringer und Patienten abgelegt. Der Header ist unabhängig von der gewählten Granularität des Dokuments; der eigentliche Inhalt des Dokuments hingegen ist abhängig vom gewählten CDA-Level: Inhalte im CDA-Level 1 sind strukturiert und enthalten Informationen wie Diagnosen, Prozeduren, weitere Behandlungsschritte, Informationen zu Anschlussterminen oder auch Freitext. Auf Ebene 2 können Einschränkungen für bestimmte klinische Dokumente gegeben werden, wie z.B. OP-Anmerkungen oder Entlassbriefe. Weitere Einschränkungen werden auf Ebene 3 vorgegeben, indem z.B. die Beschreibung von Blutdruckwerten definiert wird.

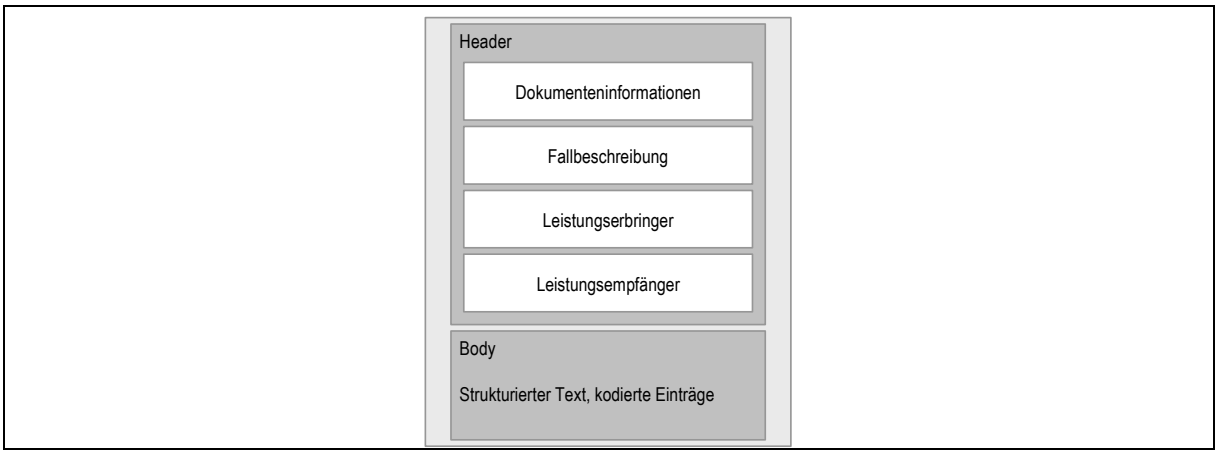

Abbildung 4.11-2: CDA-Architektur
Quelle: In Anlehnung an *Heitmann/Schweiger/Dudeck* (2003, 198)

Mit HL7 liegt ein etablierter Standard zum Austausch von klinischen Informationen und zur Strukturierung von Dokumenten vor. Die Semistrukturierung dieser Dokumente (Boeker/Jentzsch/Klar 2004, 1531) erlaubt sowohl eine automatisierte Weiterverarbeitung als auch die Visualisierung für menschliche Betrachter.

4.11.2.1.2 DICOM

DICOM (Digital Imaging and Communications in Medicine) wird für den Austausch von Daten aus Bild gebenden Verfahren eingesetzt (NEMA). DICOM zielt auf die Definition eines Standards mit den folgenden Eigenschaften ab (National Electrical Manufacturers Association 2006a, 5):

- Unabhängigkeit von medizinischen Geräten
- Grundlage für die Archivierung und Kommunikation von Bildern
- Digitale Übertragungsmöglichkeit von Bildern

Für den Austausch von Daten über ein Netzwerk werden im DICOM-Standard Protokolle definiert, die auf TCP/IP basieren (National Electrical Manufacturers Association 2006c, 7).

Für die Interoperabilität zwischen den Modalitäten werden weiterhin die Syntax und Semantik von Befehlen definiert und auch zulässige Komprimierungsverfahren wie z.B. JPEG (Joint Photographic Experts Group) angegeben (National Electrical Manufacturers Association 2006a, 15).

In Abbildung 4.11-3 ist die DICOM-Nachrichtenstruktur beschrieben (die folgenden Ausführungen sind angelehnt an die Darstellungen bei National Electrical Manufacturers Association 2006c, 10-11, 2006b, 34-36). Eine DICOM-Nachricht ist aus einer Command Set und einer Data Set zusammengesetzt, welche im Weiteren detailliert dargestellt werden:

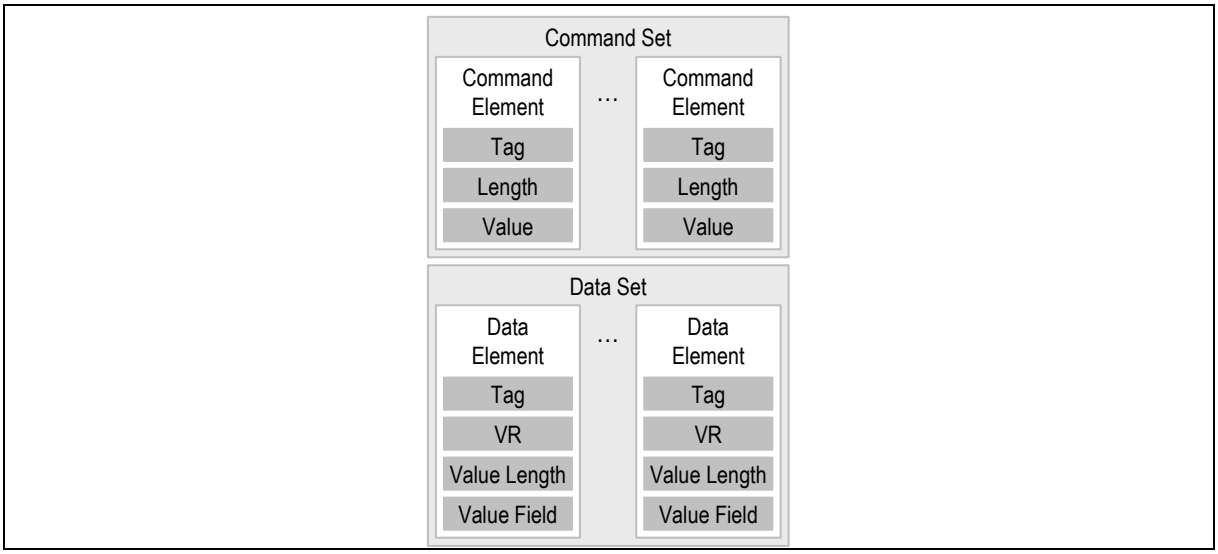

Abbildung 4.11-3: ***DICOM-Nachrichtenstruktur***
Quelle: Eigene Darstellung, in Anlehnung an *National Electrical Manufacturers Association* (2006b, 35) und *National Electrical Manufacturers Association* (2006c, 10)

Mit der Command Set wird spezifiziert, welche Operationen bzw. Nachrichten ausgeführt bzw. ausgelöst werden. Ein Command Element wird eindeutig durch ein entsprechendes Tag bestimmt. Das Element Length beinhaltet die Anzahl der Bytes, die im Feld Value gespeichert sind. Value spezifiziert den Typ des Befehls. Die möglichen Belegungen für das Command Element sind im DICOM-Standard ausführlich katalogisiert. Weil erstere für die vorliegende Arbeit nicht relevant sind, wird auf ihre Darstellung verzichtet.

Die Data Set besteht aus einer Menge von Data Elements (siehe dazu auch National Electrical Manufacturers Association 2006b, 35-36). Diese Datenelemente sind die Träger der eigentlichen Information (National Electrical Manufacturers Association 2006b, 14), d.h. Informationsobjekte (z.B. Bilder). Das Tag spezifiziert ein Gruppen- und Elementnummer. Value Representation (VR) spezifiziert den Datentyp und das Format der Werte im Value-Feld. Im Feld Value Length wird die Anzahl der Bytes des Value Field angegeben. Im Value Field werden die Werte der Datenelemente gespeichert. Die Belegungen für die Elemente werden im DICOM-Standard katalogisiert. Weil ihre Beschreibung für die vorliegende Arbeit nicht relevant ist, wird auf ihre Erläuterung im Weiteren verzichtet.

Mit den beschriebenen Eigenschaften wird DICOM oftmals als Grundlage für PACS-Applikationen (Picture Archiving and Communication System) eingesetzt. Weiterhin kann konstatiert werden, dass sich DICOM im Bereich der Bild gebenden Modalitäten etablierte. Somit ist DICOM für die genannten Anwendungsbereiche ein wesentlicher Standard und für zukünftige Informationssysteme geeignet zu berücksichtigen.

4.11.2.1.3 EDIFACT

EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) wird im administrativen Bereich zum Datenaustausch eingesetzt (United Nations Economic Commission for Europe 2005). Die ausgetauschten Daten sind nach der EDIFACT-Spezifikation strukturiert und sind damit für die automatisierte Weiterverarbeitung geeignet (United Nations Economic Commission for Europe 2005). Der Standard besitzt weltweiten Stellenwert.

Die Ausführungen in Abschnitt 4.11.2.1 zeigen, dass sich in unterschiedlichen Bereichen relevante Standards etablierten. Bis auf EDIFACT mit seinem Format für den Austausch von administrativen Daten und HL7 mit der administrativen Teilmenge sind die Standards disjunkt.

4.11.2.2 Standards im ambulanten Bereich

Im ambulanten Bereich etablierte sich in Deutschland die Standardfamilie xDT (KBV (Kassenärztliche Bundesvereinigung Deutschland)). Die folgenden Ausführungen beschreiben die unterschiedlichen xDT-Formate und sind an die Darstellungen der *KBV* (2007) angelehnt:

Ausgangspunkt für die Entwicklung der xDT-Familie war die Identifikation von Medienbrüchen bei der Abrechnung von Leistungen niedergelassener Ärzte bei der Kassenärztlichen Vereinigung (KV, KBV (Kassenärztliche Bundesvereinigung Deutschland)): In EDV-Systemen von Leistungserbringern geführte Abrechungsdaten mussten wieder ausgedruckt und in Papierform zur Abrechnung weitergeleitet werden. Mit dem von der Kassenärztlichen Bundesvereinigung definierten Standard ADT (Abrechnungsdatentransfer) konnten Abrechnungsdaten in einem einheitlichen Format über Datenträger (Disketten) mit der jeweiligen KV ausgetauscht werden. ADT wurde damit zum Standard für diesen Informationsaustausch. Weil sich ADT in der Praxis bewährte, wurde er durch weitere Derivate flankiert (für eine Zusammenfassung siehe auch KBV (Kassenärztliche Bundesvereinigung Deutschland)):

Bei der Migration von Praxisinformationssystemen stellt der Datentransfer von einem in das andere System oftmals ein Problem dar (KBV (Kassenärztliche Bundesvereinigung Deutschland)). Der Standard BDT (Behandlungsdatentransfer) definiert dazu einen Standard zum automatisierten Austausch der Behandlungsdaten. Der Austausch mit Laboren ist über die Spezifikation des Labordatenträgers (LDT) definiert. Stammdatenkostenträger (SDKT) beschreiben alle Krankenkassen und Kostenträger (KBV (Kassenärztliche Bundesvereinigung Deutschland)). Gebührenkataloge und Regelwerke für die Abrechnung werden über das Format SDGO bzw. SDRW (Stammdatei Gebührenordnung bzw. Regelwerk) von der KBV publiziert (KBV (Kassenärztliche Bundesvereinigung Deutschland)). Ein Gerätedatenträger (GDT) wird für die Übermittlung von Daten aus medizinischen Geräten eingesetzt.

Eine xDT-Nachricht ist in mehrere Abschnitte unterteilt (siehe dazu auch Abbildung 4.11-4). Die Daten werden dabei durch ASCII-Zeichen repräsentiert (KBV (Kassenärztliche Bundesvereinigung Deutschland)). Das erste Element (len) gibt die Länge der gesamten Nachricht in

Bytes an. Es folgt ein Tag, welches einem Repository entnommen ist (Heitmann/Schweiger/Dudeck 2003, 196). Letzteres ist eine Ansammlung von Tags für Datenfelder, der jeweiligen Semantik, des Typs und weiteren Einschränkungen (Heitmann/Schweiger/Dudeck 2003, 196). Im content-Element ist der eigentliche Inhalt der Nachricht gespeichert. Am Ende folgt die obligatorische Kennzeichnung des Nachrichtenendes mit den beiden Zeichen Carriage Return und Line Feed (KBV (Kassenärztliche Bundesvereinigung Deutschland)).

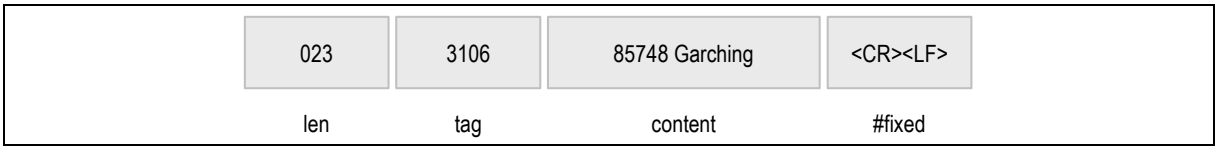

Abbildung 4.11-4: ***Beispiel für eine xDT-Nachricht***
Quelle: In Anlehnung an *Heitmann/Schweiger/Dudeck* (2003, 196)

4.11.2.3 Integration von Standards zum Datenaustausch

Die beschriebenen Standards DICOM, HL7 und xDT sind unabhängig voneinander entwickelt und deshalb nicht kompatibel zueinander (Lenz et al. 2005, 107). Der Integration unterschiedlicher Standards widmen sich die IHE- und SCIPHOX-Bemühungen, die im Folgenden beschrieben werden:

Das Ziel des IHE-Rahmenwerkes (Integrating the Healthcare Enterprise, Vegoda 2002) ist der Transfer von Informationen aus Informationssystemen innerhalb einer Institution und über die Grenzen einer Einrichtung hinweg auf der Grundlage etablierter Standards (Vegoda 2002, 22). Das IHE-Rahmenwerk besitzt dazu die folgenden Eigenschaften (die Beschreibung ist angelehnt an die Ausführungen bei Vegoda 2002, 22-23): Zur Datenweitergabe werden die Standards DICOM und HL7 in einer prozessorientierten Sicht miteinander verknüpft. Es werden weiterhin Akteure definiert, die innerhalb eines Prozesses im Rahmen von Transaktionen auf der Basis von DICOM und HL7 interagieren, um eine bestimmte Dienstleistung zu erbringen. Integrationsprofile aggregieren Akteure und Transaktionen zu einem Konglomerat zur Erbringung einer bestimmten Aufgabe innerhalb eines Prozesses.

Ursprünglich verfolgte das SCIPOHX-Projekt (Standardization of Communication between Information Systems in Physician Offices and Hospitals using XML) das Ziel, HL7 und xDT zu verknüpfen (Lenz et al. 2005, 107). Aktuell wird von der Arbeitsgruppe jedoch angestrebt, HL7-CDA an den deutschen Sprachraum anzupassen (Arbeitsgemeinschaft SCIPHOX GbR mbH) und die Kommunikation zwischen dem ambulanten und stationären Bereich zu unterstützen. Mit dieser Anpassung erfolgte einerseits die Übersetzung und Interpretation der CDA-Header-Tags und des entsprechenden Sprachschatzes in den deutschen Sprachraum, andererseits die Erweiterung zur Berücksichtigung von lokalen Eigenschaften wie Informationen zum Versichertenverhältnis des Patienten (Heitmann/Schweiger/Dudeck 2003, 199). In der Phase 1 des Projektes liegt der Fokus auf der Definition von Entlassbriefen und Überweisungsschreiben, die jeweils auf der Basis von HL7-CDA entworfen werden (Heitmann/Schweiger/Dudeck 2003, 197). In Phase 2 wird eine Erweiterung z.B. hinsichtlich der Unterstützung des elektronischen Rezepts, der Berücksichtigung von Sicherheit beim Trans-

port der Daten oder spezieller Domänen wie Krebs- oder Diabetes-Patienten angestrebt (Heitmann/Schweiger/Dudeck 2003, 202).

Aus den Beschreibungen in Abschnitt 4.11.2.3 kann gefolgert werden, dass sich etablierte Standards mit jeweils unterschiedlichen Zielen zusammenführen bzw. auf nationale Gegebenheiten anpassen lassen. Somit lassen sich die jeweiligen Vorteile vereinen bzw. auf andere Rahmenbedingungen übertragen.

4.11.3 Standards zur Terminologie

Unterschiedliche Terminologien erschweren die Kommunikation zwischen Leistungserbringern auf semantischer Ebene. Dies trifft nicht nur auf die digitale Kommunikation, sondern auch auf andere Kommunikationsformen zu. Um eine Basis für ein gemeinsames Verständnis der Information zu schaffen, helfen terminlogische Standards. Einerseits ermöglicht die einheitlich definierte Semantik eine Interoperabilität zwischen den am Behandlungsprozess beteiligten Personen. Die menschlichen Akteure tauschen ihre Ergebnisse über einen gemeinsamen terminologischen Standard aus. Damit können mögliche Fehlerquellen reduziert werden, die durch die unterschiedliche semantische Belegung und die daraus resultierende Interpretation von Termini verursacht werden. Andererseits ermöglicht eine einheitliche Festlegung einer semantischen Basis die automatische Verarbeitung von Daten, um z.B. Informationen zu Forschungszwecken zu durchsuchen oder entsprechend dem informationslogistischen Prinzip aufzubereiten. Etablierte Standards für diesen Bereich werden in den folgenden Abschnitten 4.11.3.1 mit 4.11.3.5 beschrieben.

4.11.3.1 LOINC

LOINC (Logical Observation Identifiers Names and Codes, Regenstrief Institute Inc. 2004a) ist ein Code-System, welches den darin aufgenommenen medizinischen Begriffen aus dem Labor- und klinischen Bereich (Regenstrief Institute Inc. 2004a) eindeutige Codes zuweist. Die LOINC-Datenbank umfasst zu diesem Zweck mehrere Kategorien und Einträge für die zu codierenden Begriffe, wobei zum Zeitpunkt der Erstellung der vorliegenden Arbeit ca. 41.000 Einträge dieser der Datenbank vorliegen (Regenstrief Institute Inc. 2004b). Diese Einträge enthalten weitere Informationen wie z.B. einen formalen Namen oder Synonyme (Regenstrief Institute Inc. 2004b) zu dem jeweiligen Begriff.

4.11.3.2 ICD

Mit ICD-10 (International Classification of Diseases, World Health Organization 2006) wird von der World Health Organization ein internationaler Standard zur Klassifikation von Krankheiten gepflegt. Die aktuelle Version trägt die Kennzeichnung ICD-10 und bezeichnet die zehnte Revision der Klassifikation. Jährlich erfolgt eine geeignete Anpassung entsprechend der jeweils aktuellen Situation im Gesundheitswesen. Weiterhin bestehen nationale Anpassungen der Kategorisierung.

Mit einer mehrstufigen Klassifikation wird die genaue Beschreibung, ausgehend von einer grob granularen Zuordnung, immer weiter eingegrenzt. Ähnlich einer Baumstruktur wird an den Knotenstellen die passende Alternative für die weitere Spezifikation gewählt, bis ein Blatt erreicht wird, dem keine weiteren Knoten folgen.

4.11.3.3 UMLS

UMLS (Unified Medical Language System®, United States National Library of Medicine 2005) zielt auf die automatische Verarbeitung der Semantik aus den Bereichen Gesundheit und Biomedizin ab. Dazu wird die Konsolidierung unterschiedlicher Standards zur Etablierung eines einheitlichen Wortschatzes fokussiert, indem verschiedene Vokabulare, Codiersysteme oder Klassifikationen wie LOINC, HL7, ICD-10 oder SNOMED CT (siehe Abschnitt 4.11.3.4) zusammengeführt werden.

UMLS umfasst die Bestandteile Metathesaurus®, Semantic Network und SPECIALIST Lexicon (United States National Library of Medicine 2004). Im Folgenden werden diese Elemente, angelehnt an die Ausführungen bei *United States National Library of Medicine*® (2004), beschrieben:

Im Metathesaurus werden Konzepte, ihre Bezeichnungen und Beziehungen in mehreren Sprachen beschrieben. Die beinhalteten Konzepte stehen über zwei Relationen zueinander in Beziehung: Gleiche Konzepte aus den Quellvokabularen werden verbunden, um unterschiedliche Bezeichnungen für gleiche Begriffe zu vereinen (Bodenreider et al. 1998, 815). Unterschiedliche Konzepte sind über Beziehungen verbunden, um jeweils sinnvolle Assoziationen zu identifizieren.

Jedes Konzept aus dem Metathesaurus ist mit mindestens einem semantischen Typ aus dem Semantic Network assoziiert. Damit wird eine Klassifizierung aller Konzepte auf der abstrakten Ebene des Semantic Network erreicht. Zusätzlich zu dieser Kategorisierung werden auch Beziehungen zwischen den im Semantic Network enthaltenen Klassen etabliert. Im Semantic Network werden zudem die Klassen durch Text und hierarchische Beziehungen beschrieben. Konzepte im Semantic Network werden durch Knoten und Beziehungen durch Kanten in einem Graph dargestellt.

Im SPECIALIST Lexicon werden zu jedem Eintrag syntaktische, morphologische und orthografische Informationen hinterlegt.

4.11.3.4 SNOMED

SNOMED CT (the Systemazited Nomenclature of Medicine Clinical Terms, SNOMED International 2005) zielt auf die Vereinbarung einer einheitlichen Terminologie für medizinische Begriffe ab und umfasst die Unterstützung für die Sprachen Englisch, Deutsch und Spanisch. Mit dieser Nomenklatur werden also sowohl ein allgemeingültiger Rahmen als auch die Anpassung für länderspezifische Erweiterungen unterstützt (SNOMED International o.J.-a). SNOMED CT ist skalierbar und damit in der gewählten Abstraktionsebene anpassbar entsprechend den Anforderungen für die Verwendung z.B. in Kliniken durch das jeweilige Personal oder für den Einsatz bei Kostenträgern (SNOMED International o.J.-a).

Um die Interoperabilität mit bereits eingesetzten Klassifikationen zu erreichen, werden Abbildungen zwischen Begriffen aus SNOMED CT und z.B. ICD-10 definiert. Weiterhin wird die

Verknüpfung zu Standards aus anderen Bereichen als der Terminologie wie z.B. HL7 realisiert (SNOMED International 2005).

Weitere SNOMED CT-Eigenschaften lassen sich in Anlehnung an die Ausführungen bei *SNOMED International* (o.J.-b) wie folgt zusammenfassen: SNOMED CT umfasst 18 hierarchisch strukturierte Hauptkonzepte. Beziehungen zwischen diesen Konzepten werden durch Relationen abgebildet, wobei zwischen IS-A-Relationen zur Verbindung von Konzepten innerhalb derselben Hierarchie und Attributrelationen zur Verbindung von Konzepten zwischen verschiedenen Hierarchien differenziert wird. Die einzelnen Konzepte tragen Identifikatoren und vollständige Beschreibungen.

4.11.3.5 GALEN

GALEN (Generalised Architecture for Languages, Encyclopæidas and Nomenclatures in Medicine, Rector/Nowlan 1994) zielt auf die Entwicklung eines sprachunabhängigen Konzeptes zur Modellierung von medizinischen Klassifikationen und Codes (Rector/Nowlan 1994, 75) ab. Die folgenden Ausführungen fassen die Darstellung bei *Rector/Nowlan* (1994, 75) zusammen:

Im Zentrum von GALEN steht die formale Modellierungssprache GRAIL (GALEN Representation and Integration Language). Dabei wird strikt zwischen den Modellen und den linguistischen Mechanismen getrennt, welche die Modelle interpretieren. Mit dieser Aufteilung wird erreicht, dass das Modell vollständig unabhängig von der jeweiligen natürlichen Sprache ist. Mit GRAIL werden damit formale Konstrukte erstellt, die verifizierbar sind und keine Widersprüche sowie Ambiguitäten enthalten. Ein weiterer Bestandteil von GALEN ist ein Referenzmodell für Codes (Coding Reference Model, CORE) zur medizinischen Terminologie, welche z.B. Bereiche der Pathologie, Anatomie oder Therapie abdecken.

Die modellierten Klassifikationen werden auf einem Terminologie-Server (TeS) zur Verfügung gestellt und in der Form des OpenGALEN-Projektes (OpenGALEN) bereitgestellt.

Die beschriebenen terminologischen Standards besitzen jeweils unterschiedliche Schwerpunkte. Daraus lässt sich ihre Diversifizierung ableiten. Weil der Integration dieser Standards in der integrierten Versorgung eine wesentliche Rolle zukommt, kann erstere zumindest ansatzweise bei UMLS und SNOMED CT festgestellt werden. Die Integration dieser Standards kann angesichts der Vielzahl der existierenden Ansätze im Vergleich zur Etablierung einer einzigen und einheitlichen Terminologie als die anzustrebende Alternative betrachtet werden.

4.11.4 Standards für Dokumente

Neben Standards für den Datenaustausch sowie die Interpretation von Inhalten sind Dokumentenstandards für die integrierte Versorgung relevant. Dazu wird die gesamte Struktur für umfangreiche Datenaggregationen definiert. Ein Beispiel für einen Dokumentenstandard ist openEHR und wird deshalb im Folgenden beschrieben wird:

openEHR (open Electronic Health Record, openEHR 2004) entstand aus dem GEHR-Projekt (Good European Health Record, Good European Health Record 1995; GEHR 2006) und ist eine unabhängige und offene Initiative (The openEHR Foundation 2005, 1) mit dem Ziel der Entwicklung einer offenen, interoperablen Plattform, deren wesentliche Komponente eine kli-

nisch effektive und interoperable elektronische Gesundheitsakte ist (The openEHR Foundation 2005, 2). Dazu werden klinische Anforderungen erhoben, Spezifikationen erstellt und Implementierungen konstruiert, wobei die folgenden Anforderungen fokussiert werden (The openEHR Foundation 2005, 2):

- Dokumentation aller klinischen Informationen
- Anpassungsmöglichkeit des Inhalts, der Semantik und der Benutzerschnittstellen unabhängig vom jeweils eingesetzten Informationssystem
- Integration etablierter Terminologiesysteme wie SNOMED-CT, LOINC, ICD und ICPC[27]
- Integration der Nachrichtensysteme HL7 (Version 2) und EDIFACT
- Integration anderer Krankenhausinformationssysteme und Datenbanken
- Integration von Applikationen über eine API
- Verteilte Versionierung und Zusammenführung der gespeicherten Informationen
- Komponentenbasierte, adaptive und zukunftssichere Gestaltung der Architektur

openEHR stellt zwar einen offenen Standard zur Definition von Dokumenten zur Verfügung, dennoch ist der Schwerpunkt auf den klinischen Bereich beschränkt. Deshalb wird zwar innerhalb eines Sektors der standardisierte Dokumentenaustausch ermöglicht, es fehlt aber die Integration aller Sektoren des Gesundheitswesens. Für die integrierte Versorgung ist deshalb eine geeignete Dokumentenstruktur zu wählen.

4.12 Anforderungskatalog für ein IT-System im Gesundheitswesen

Um die Fülle der Anforderungen, die sich aus den bisherigen Ausführungen zum Gesundheitswesen in Abschnitt 4 ableiten lassen, zu beherrschen, ist ihre Kategorisierung erforderlich. Weil bereits einzelne Institutionen mit dem Integrationsproblem konfrontiert (siehe Abschnitt 4.6.1) sind und im angestrebten Anforderungskatalog nicht zwischen Informationssystemen innerhalb einer Institution und solchen, die sich über deren Grenzen hinaus erstrecken, differenziert werden wird, wird im Folgenden eine Klassifizierung von Integrationsanforderungen auf der Basis des in Abschnitt 4.9.4.6 beschriebenen Stufenmodells vorgenommen. Dieses Modell wird um die zusätzliche Stufe Anwendungsebene und eine Umgebung ergänzt, die die besonderen Charakteristika des Gesundheitswesens reflektieren (siehe Abbildung 4.12-1).

Diese Aufteilung berücksichtigt somit ein erweitertes Verständnis eines Informationssystems, welches als ein soziotechnisches System (Wissenschaftliche Kommission der Wirtschaftsinformatik (WKWI) 1994, 80) nicht nur menschliche, in dem vorliegenden Fall also die Akteure des Gesundheitswesens, und technische Komponenten, sondern insbesondere auch übrige Eigenschaften der Domäne Gesundheitswesen betrachtet.

[27] Für weitere Informationen siehe http://www.globalfamilydoctor.com, zugegriffen am 30.09.2006.

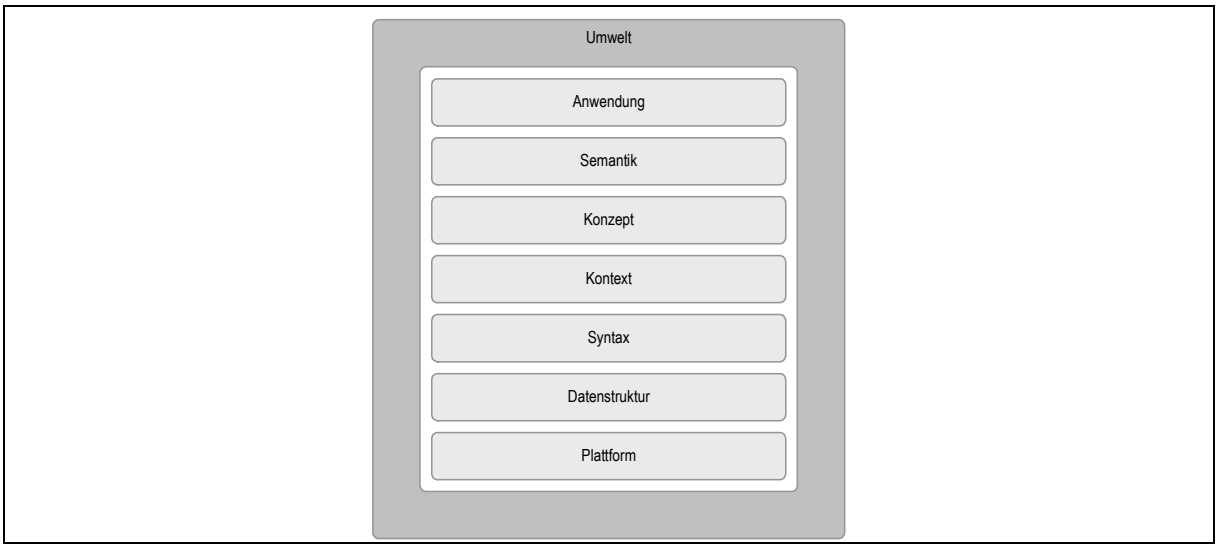

Abbildung 4.12-1: Kategorisierung des Anforderungskatalogs
Quelle: Eigene Darstellung, in Anlehnung an *Lenz et al.* (2005, 115)

4.12.1 Plattform

In den Abschnitten 4.10 und 4.11.1 wurden mehrere Kommunikationsstrukturen beschrieben. Obwohl die im Entstehen befindliche TI für die eGK die zentrale Plattform darstellen wird und damit andere Plattformen obsolet werden, konnten doch aus den bisherigen Projekten zur Etablierung einer Kommunikationsinfrastruktur (siehe dazu Abschnitt 4.11.1) Erfahrungswerte gesammelt werden, die für die Konstruktion der TI wertvoll sind und dort einfließen.

Um das Funktionalitätspotenzial vollständig auszuschöpfen, ist ein Informationssystem so anzulegen, dass es in eine Infrastruktur eingebettet werden kann, um somit die Verteilung im Gesundheitswesen zu unterstützen. Weiterhin ist auch zu berücksichtigen, dass ein Informationssystem möglichst auf einer plattformunabhängigen Laufzeitumgebung implementiert wird. Dies ist insbesondere im Kontext von heterogenen Hardware-Plattformen erforderlich. Im weiteren Verlauf der vorliegenden Arbeit wird herausgearbeitet, dass sich Agentensysteme hinsichtlich erreichbarer Flexibilität für Integrationsszenarien besonders vorteilhaft einsetzen lassen. In Anforderung 1 lassen sich die genannten Eigenschaften für ein adäquates Informationssystem zusammenfassen:

Anforderung 1: Integration in eine geeignete Kommunikationsplattform

4.12.2 Datenstruktur

Weil sich die Domäne Gesundheitswesen ständig weiterentwickelt (siehe auch Nadkarni et al. 1999, 478), ist eine entsprechende Datenstruktur erforderlich, die dieser Anforderung genügt. Dazu beschreiben *Nadkarni et al.* (1999) einen geeigneten Ansatz, um diese Voraussetzung zu erfüllen (siehe Abschnitt 4.9.4.1). Weil damit bereits ein tragfähiges Konzept zur Problemlösung besteht und der Fokus dieser Arbeit nicht auf dieser Ebene angesiedelt ist, wird diese Anforderung bei der Zusammenstellung des Anforderungskatalogs nicht berücksichtigt.

4.12.3 Syntax

Sollen Daten zwischen Informationssystemen ausgetauscht werden, folgen diese definierten Strukturen bzw. einer vereinbarten Syntax. Deshalb ist eine wesentliche Voraussetzung für ein Informationssystem die Berücksichtigung von syntaktischen Standards (Haas 2005d, 91). Auch diese Anforderung, insbesondere die Transformation unterschiedlicher syntaktischer Standards ist aus technischer Sicht bereits gegeben (Lenz et al. 2005, 114). Dennoch wird in der vorliegenden Arbeit demonstriert, wie syntaktische Standards in ein agentenbasiertes System integriert werden können, um auf deren Basis eine virtuelle Patientenakte als Integrationsträger zu konstruieren. Deshalb wird in Anforderung 2 die Unterstützung von syntaktischen Standards berücksichtigt:

Anforderung 2: Unterstützung von syntaktischen Standards

4.12.4 Kontext

Für die Zusammenführung der Daten in einer Patientenakte ist eine eindeutige und Institutionen übergreifende Patientenidentifikation (Lenz et al. 2005, 116) oder eine geeignete Verknüpfung Institutionen bezogener Identifikationen (Wang/Ohe 1999) erforderlich. Erstere wird mit der Einführung der eGK gegeben sein und wird deshalb in der vorliegenden Arbeit fokussiert. Auf der Basis dieser Information lässt sich eine einrichtungsübergreifende und virtuelle Patientenakte aggregieren. Somit ist eine eindeutige Patientenidentifikation eine wesentliche Voraussetzung für die integrierte Versorgung und wird somit als Anforderung 3 bestimmt:

Anforderung 3: Bereitstellung einer eindeutigen Patientenidentifikation

4.12.5 Konzept

Auf Konzeptebene ist die „Kompatibilität auf Typebene“ (Lenz et al. 2005, 114) erforderlich. Auch dafür wurden in Abschnitt 4.9.4.3 bereits Lösungsansätze beschrieben, die die Integration auf Konzeptebene zumindest bis zu einem gewissen Grad realisieren. Zudem fokussiert die vorliegende Arbeit nicht diese Problemstellung. Deshalb wird im Folgenden darauf verzichtet, dieses Kriterium als Anforderung für ein integriertes Informationssystem zu formulieren.

4.12.6 Semantik

Damit Daten nicht nur maschinell, sondern auch durch Menschen eindeutig weiterverarbeitet werden können, ist eine Konsolidierung des medizinischen Wissens auf semantischer Ebene (Haas 2005d, 91) erforderlich. Dazu existiert eine Reihe von etablierten Standards, die in Abschnitt 4.11.3 beschrieben wurden. Da die dargestellten Standards oftmals unterschiedliche Schwerpunkte fokussieren, muss ein Informationssystem in der Lage sein, semantische Standards geeignet zu verarbeiten. Dieses Kriterium wird deshalb als Anforderung 4 formuliert:

Anforderung 4: Unterstützung von semantischen Standards

4.12.7 Anwendung

Auf Anwendungsebene wurde bei den bisherigen Ausführungen in Abschnitt 4 ein sehr umfangreiches Portfolio an Potenzialen identifiziert. Für dieses wird zunächst ein geeignetes Klassifizierungskonzept eingeführt. Dieses stützt sich auf die Anforderungsanalyse bei *Lenz et al.* (2005, 110-112). Demnach lassen sich die Anwendungssäulen Informationslogistik, Entscheidungs- und Prozessunterstützung als Anforderung 5 identifizieren. Dazu sind jeweils geeignete Architekturen für Informationssysteme zu identifizieren.

Anforderung 5:	*Anwendungssäulen Informationslogistik, Entscheidungs- und Prozessunterstützung*

4.12.8 Charakteristika der Domäne

Die in Abschnitt 4 analysierten Charakteristika der Domäne werden in Anlehnung an die Anforderungserhebung bei *Lenz et al.* (2005, 110-111) in die Bereiche Informationsmedien, Flexibilität, Unterstützung der Kommunikation, Systemnutzung und Patientenorientierung aufgeteilt sowie in Anforderung 6 formuliert:

Anforderung 6:	*Domänenbereiche Informationsmedien, Flexibilität, Unterstützung der Kommunikation, Systemnutzung und Patientenorientierung*

4.13 Zusammenfassung

Das Gesundheitswesen zeichnet sich durch besondere Eigenschaften aus, die eine geeignete Unterstützung durch IT-Systeme erschweren. Die umfangreiche Anzahl von nationalen und internationalen Ansätzen zur Etablierung von geeigneter IT-Unterstützung im Gesundheitswesen untermauert den Komplexitätsgrad. Die Erfahrungen aus diesen Projekten können in die finale Spezifikation und spätere Weiterentwicklung der aktuell im Entstehen befindlichen TI für die eGK einfließen. Diese Infrastruktur bietet ein wesentliches Potenzial, insbesondere die Informationslogistik und die Patientenzentrierung zu unterstützen. Damit werden Behandlungen nicht mehr fokussiert auf einzelne Abteilungen oder Institutionen betrachtet, sondern vielmehr als ein Prozess, der sich an der Genesung des Patienten ausrichtet und damit die Grenzen von Abteilungen oder Einrichtungen überschreiten kann. Eine Orientierung des Behandlungsprozesses am Patienten induziert somit eine Prozessreorganisation zu einer ganzheitlichen Versorgung des Patienten. Weiterhin werden mit der Einführung der TI Effizienzsteigerungen und damit eine Erhaltung der Behandlungsqualität bei einer Kostenreduktion bzw. einer Stagnation der Ausgaben erwartet.

Die eGK und die mit ihr etablierte Infrastruktur können als Wegbereiter für die elektronische Patientenakte betrachtet werden. Auf der Basis der TI sind neben den vorgesehenen obligatorischen und freiwilligen Anwendungen auch weitere Mehrwertdienste denkbar (Neuhaus/Deiters/Wiedeler 2006). Die Einführung der eGK bietet den Institutionen im Gesundheitswesen insbesondere das Potenzial, ihre Architekturen im Vorfeld zu reflektieren und geeignet anzupassen, um vor allem aus den freiwilligen Anwendungen Nutzen zu ziehen (siehe dazu die Ergebnisse der Studie von Bernnat/Booz Allen Hamilton GmbH 2006, 24 und 27 sowie 195-264).

Aus der Darstellung des Status quo im Gesundheitswesen und der teilw. daraus ableitbaren Anforderungen für ein geeignetes Informationssystem wurde ein abstrakter und im Folgenden weiter zu konkretisierender Anforderungskatalog abgeleitet. Damit ist Forschungsfrage 1 (siehe Abschnitt 1.4) beantwortet.

5 Multiagentensysteme im Gesundheitswesen

5.1 Überblick

Die Ausführungen in Abschnitt 2 beschrieben Software-Agenten und ihre potenzielle Vorteilhaftigkeit gegenüber herkömmlichen Programmiertechniken unter bestimmten Voraussetzungen. In Abschnitt 3 folgte eine Darstellung von Architekturen, wobei die Potenziale von agentenbasierten Architekturen ohne einen besonderen Bezug zu einer bestimmten Domäne aufgezeigt wurden. In Abschnitt 4 wurden wesentliche Eigenschaften des Gesundheitswesens sowie Anforderungen an unterstützende Informationssysteme beschrieben, die in einem Anforderungskatalog verdichtet wurden.

Auf der Basis dieser Vorarbeiten (Abbildung 4.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1) folgt in diesem Abschnitt 5 eine Abbildung von Eigenschaften der Domäne Gesundheitswesen auf geeignete Konstrukte der Software-Technik.

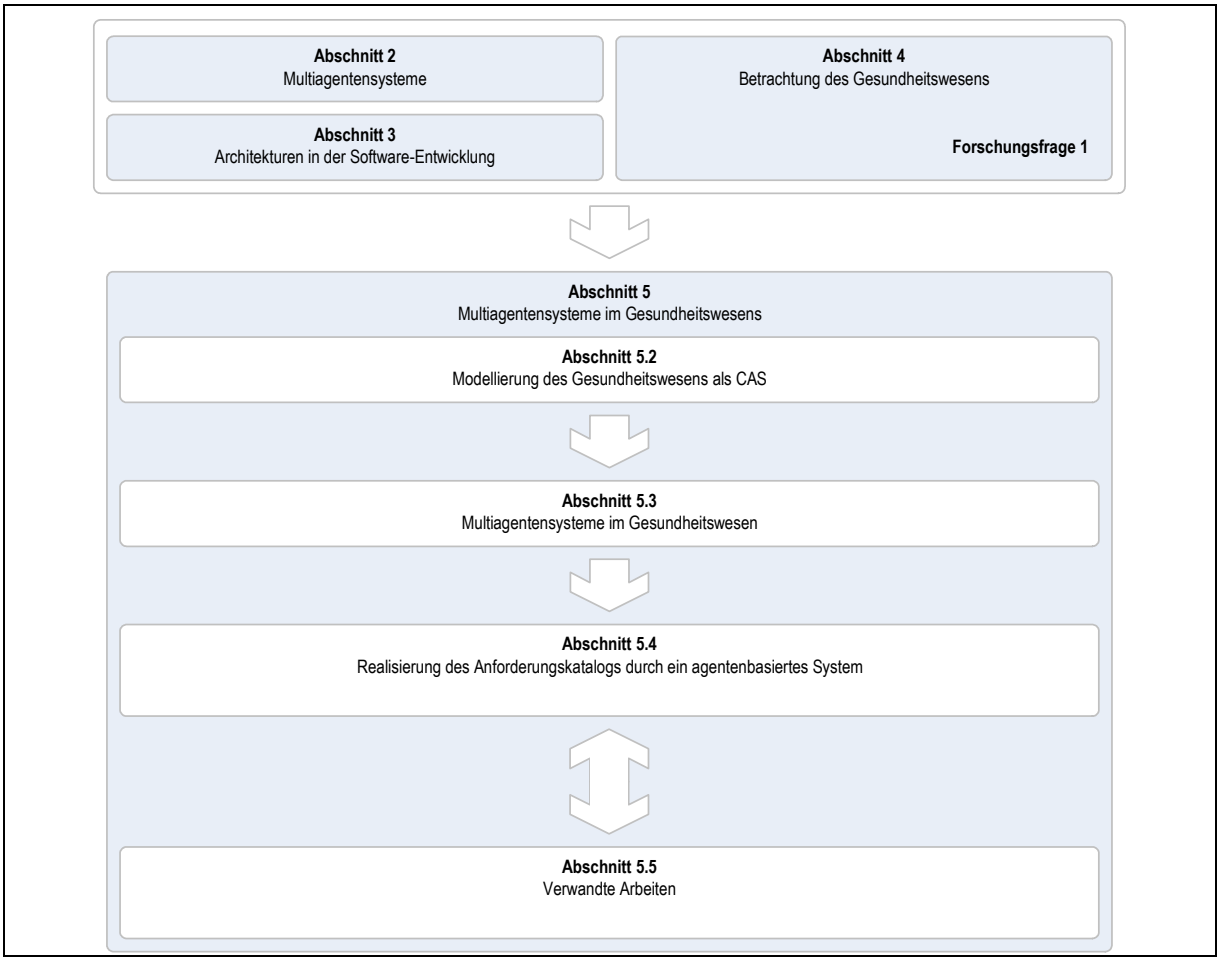

Abbildung 5.1-1: ***Überblick über Abschnitt 5***
Quelle: Eigene Darstellung

Dazu wird die Modellierung des Gesundheitswesens als CAS in Abschnitt 5.2 dargestellt. Weil sich die wesentlichen Eigenschaften eines CAS auf agentenbasierte Systeme abbilden

lassen (Abschnitt 5.3), kann über eine transitive Relation für das Gesundheitswesen auf der Basis von Software-Agenten ein geeignetes Informationssystem konstruiert werden. In Abschnitt 5.4 wird beschrieben, wie der in Abschnitt 4.12 dargestellte Anforderungskatalog für Informationssysteme im Gesundheitswesen durch ein agentenbasiertes System realisiert werden kann. Um das Alleinstellungsmerkmal dieser Darstellung herauszuarbeiten, werden in Abschnitt 5.5 verwandte Arbeiten beschrieben.

5.2 Modellierung des Gesundheitswesens als CAS

Im Folgenden werden, ausgehend von der Darstellung des Systembegriffes, diejenigen Eigenschaften eines CAS sowie die Notwendigkeit für die Modellierung des Gesundheitswesens als CAS beschrieben:

Ein System kann als eine Menge von Komponenten und Interaktionen mit einer Zieldefinition definiert werden (Plsek 2001, 309). Übertragen auf das Gesundheitswesen bedeutet dies, dass die Akteure bzw. Einrichtungen (Komponenten) interagieren (Interaktionen), um die Gesundheit (Ziel) eines Patienten zu erhalten oder wiederherzustellen (Plsek 2001, 309). Weil aber im Gesundheitswesen Dynamik und Nichtdeterminismus beobachtet werden können (siehe Abschnitt 4.2.2.2), ist eine zentrale Steuerung wie in einem vollständig spezifizierten System, das zur Ausführungszeit seiner Spezifikation folgt, welche zur Entwicklungszeit determiniert wurde und somit antizipierbar ist, meist nicht sinnvoll (siehe auch Plsek 2001, 309-312). Deshalb wird vorgeschlagen, das Gesundheitswesen als CAS zu betrachten (Committee on Quality of Health Care in America: Institute of Medicine 2001, 63-88; Plsek 2001, 312; Tan/Wen/Awad 2005).

Ein solches System ist definiert als eine Ansammlung von individuellen Einheiten, die sich so verhalten können, wie es nicht notwendigerweise antizipierbar ist, und deren Aktionen derart miteinander verknüpft sind, dass sich diese auf den Kontext der anderen Einheiten auswirken (Plsek 2001, 313). *Plsek* (2001, 313) weist im Zusammenhang mit komplexen, adaptiven Systemen insbesondere auf die Arbeiten von *Wilson* (1971), *Varela/Coutinho* (1991), *Lewin* (1992), *Waldrop* (1992), *Wheatley* (1992), *Kelly* (1994), *Gell-Mann* (1995), *Morowitz/Singer* (1995), *Stacey* (1996), *Brown/Eisenhardt* (1998), *Zimmermann/Lindberg/Plsek* (1998) und *Mandelbrot* (1999) hin und zeigt damit, dass das CAS unterschiedlichen Disziplinen entstammt. Komplexe, adaptive Systeme besitzen nach *Plsek* (2001, 313-314) die folgenden detaillierten Eigenschaften:

- *Adaptive Elemente*: Die Elemente des Systems können sich aus sich heraus ohne einen direkten Einfluss verändern.
- *Einfache Regeln*: Einfache Regeln, die lokal angewendet werden, führen zu einem emergenten, komplexen Systemverhalten.
- *Nichtlinearität*: Der Zusammenhang zwischen Aktionen und ihren Ergebnissen muss nicht notwendigerweise linear sein. Kleine Veränderungen können zu umfangreichen Reaktionen führen.
- *Emergentes Verhalten* und *Neuheit*: Das System ist geprägt von kontinuierlicher Kreativität.

- Keine detaillierte *Vorhersagbarkeit*: In einem adaptiven System können detaillierte Vorhersagen nur mit einem gewissen Grad an Unzuverlässigkeit getroffen werden. Wie sich ein System im Detail verhält, kann nur durch dessen Beobachtung beurteilt werden. Dies ist mit der Adaptivität, der Nichtlinearität sowie der Emergenz zu begründen.
- *Inhärente Steuerung*: Die Steuerung des Systems erfolgt durch Selbstorganisation[28].
- *Kontext* und *Einbettung*: Systeme besitzen einen Kontext, der wiederum durch ein weiteres System bestimmt sein kann. Kontexte beeinflussen andere Systeme.
- *Koevolution*: Ein komplexes, adaptives System wird von permanenter Spannung und Entspannung getrieben.

Bei der Betrachtung von Eigenschaften des Gesundheitswesens aus der Perspektive eines CAS arbeiten *Tan/Wen/Awad* die folgenden Aspekte heraus und begründen damit die Notwendigkeit der Modellierung des Gesundheitswesens als CAS (2005, 39-41):

Die Komplexität im Gesundheitswesen ist bedingt durch die große Anzahl der Beteiligten und den hohen Grad ihrer Spezialisierung, womit die Aggregation von sehr spezifischem Fachwissen auf Informationsinseln einhergeht. Neben den Akteuren steht der Patient im Vordergrund. Da jedoch der menschliche Körper mit seinen Funktionen ein diffiziles Konstrukt darstellt, ergeben sich wegen unterschiedlicher Behandlungen und Reaktionen des Patienten mannigfaltige Behandlungsprozesse.

Komplexe Interaktionen lassen sich z.B. in den Bereichen der Kooperation, Kommunikation und Koordination der Akteure im Leistungserbringungsprozess identifizieren. Bspw. sind bei einer Terminvereinbarung die Einschränkungen meist mehrerer Akteure zu berücksichtigen. Terminänderungen, induziert z.B. durch nicht antizipierbare Notfälle oder weitere Abhängigkeiten, die dann z.B. gegeben sind, wenn Voraussetzungen eines Patienten für eine Operation nicht erfüllt sind, haben Auswirkungen auf die bereits vereinbarten Terminpläne anderer Patienten oder des medizinischen Personals. Damit wirken sich Änderungen auf den Kontext anderer Einheiten aus.

Die Adaptivität in einem CAS ist gegeben durch die Akteure im Gesundheitswesen, die sich ständig verändern (Committee on Quality of Health Care in America: Institute of Medicine 2001, 64). Somit sind strikte Behandlungsabläufe ohne Berücksichtigung von sich dynamisch ändernden Bedingungen weniger adäquat, um alle Eventualitäten in einem Behandlungsprozess abzudecken (Tan/Wen/Awad 2005, 39).

Die Ausführungen zeigen, dass das Gesundheitswesen als ein komplexes, adaptives System modelliert werden kann, wobei die wesentlichen Eigenschaften des Gesundheitswesens im Sinne eines CAS wie folgt zusammengefasst werden können:

- *Hohe Anzahl von Akteuren*
- *Inhärent komplexe Interaktionen*
- *Dynamik und Adaptivität*

[28] *Plsek* (2001, 313-314) verweist im Zusammenhang mit dem Begriff der Selbstorganisation auf die Arbeiten von *Kauffmann* (1995), *Holland* (1998) und *Prigogine* (1967; 1980).

In dem folgenden Schritt gilt es nun, das CAS-Modell des Gesundheitswesens auf geeignete Konstrukte und Architekturen der Software-Technik abzubilden, um die Grundlage für die Implementierung eines Informationssystems abzuleiten. Dieser Zusammenhang wird in den folgenden Abschnitten 5.3.1 und 5.3.2 beschrieben.

5.3 Multiagentensysteme im Gesundheitswesen

5.3.1 Abbildung des Gesundheitswesens auf den Agentenbegriff

Software-Agenten sind geeignet, das Modell eines komplexen, adaptiven Systems zu implementieren. Dies wird im Folgenden über die Determinanten Anzahl von Akteuren, Interaktionen sowie Dynamik und Adaptivität (siehe Abschnitt 5.2) begründet:

Die umfangreiche Anzahl der im Gesundheitswesen agierenden Akteure lässt sich jeweils mit dedizierten Agenten mit geeigneten Rollen abbilden. Damit können die Akteure unmittelbar als Konstrukte der Software-Technik modelliert werden. Auch aus technischer Perspektive ist diese Abbildung möglich: Die Skalierbarkeit eines umfangreichen Agentensystems konnte bei *Chmiel et al.* (2005) bereits nachgewiesen werden.

Multiagentensysteme (MAS) sind wesentlich durch spezifizierte Interaktionen ihrer Agenten gekennzeichnet. Damit kann die Komplexität der Interaktionsmuster bzw. ihre Realisierung von der übrigen Funktionalität eines Agenten separat betrachtet werden, womit die Implementierungskomplexität reduziert wird. Diese Interaktionen können dabei in dem breiten Spektrum von simplen Methodenaufrufen bis hin zu lose gekoppelten Nachrichten variieren (siehe auch Jennings 2001, 37). Komplexe Interaktionen können in einem Agentensystem geeignet durch spezielle Kommunikationsprotokolle[29] abgebildet werden.

Ein MAS kann weiterhin Eigenschaften eines emergenten Systems besitzen (siehe auch Abschnitt 3.3.4). Dies ist in einem entsprechenden Informationssystem im Gesundheitswesen z.B. bedingt durch den zu Beginn einer Behandlung noch nicht fest stehenden Behandlungsprozess und unvorhersehbar eintretende Prioritätsverschiebungen von Personal und Ressourcen durch Notfälle (Kirn et al. 2000). Bei nicht antizipierbaren Ereignissen wie Notfällen kann ein Agentensystem geeignet reagieren, indem z.B. adäquate Terminverschiebungen angestoßen werden. Damit ist das Verhalten der Komponenten eines Systems nicht im Detail antizipierbar. Zusätzlich können durch Terminverschiebungen weitere Änderungen z.B. im Gerätemanagement der für bestimmte Patienten bereitgestellten medizinischen Geräte ausgelöst werden. Somit ändert sich wiederum der Kontext anderer Patienten, weshalb auch hier ggf. Terminverschiebungen erforderlich sind. Bisherige wissenschaftliche Erkenntnisse (Kirn 2006) bestätigen, dass Software-Agenten insbesondere bei der Erfüllung dieser flexiblen Anforderungen vorteilhaft eingesetzt werden können.

Mit diesen Ausführungen (siehe dazu auch Abschnitt 5.1) lässt sich ein CAS entsprechend den Darstellungen bei *Tan/Wen/Awad* (2005) als ein MAS modellieren, denn die wesentlichen Eigenschaften eines CAS können adäquat durch Agenten abgebildet werden. Auch weitere Gründe sprechen für die Abbildung des Gesundheitswesens durch MASe:

[29] Dazu wird von der FIPA ein Satz von Protokollen definiert. Deren Spezifikationen können auf der folgenden Seite abgerufen werden: http://www.fipa.org/repository/standardspecs.html, zugegriffen am 20.02.2007.

Das medizinische und pflegerische Personal agiert an unterschiedlichen Orten mit verschiedenen Absichten und Bedürfnissen. Somit kann im Gesundheitswesen ein potenzielles Einsatzgebiet der ambienten Intelligenz (siehe Abschnitt 3.3) identifiziert werden: Agenten können im Sinne der ambienten Intelligenz als ein persönlicher Assistent der jeweiligen Akteure fungieren und dementsprechend z.B. relevante Inhalte nach dem informationslogistischen Prinzip und proaktiv zur Verfügung stellen. Insbesondere die Proaktivität ist es, die bisherigen Informationssystemen im Gesundheitswesen nicht attribuiert werden kann (für weitere Ausführungen zur Notwendigkeit der Einführung von proaktiven im Gegensatz zu reaktiven Informationssystemen im Gesundheitswesen siehe auch Zachewitz 2004b). Die technischen Voraussetzungen zur Realisierung der persönlichen Assistenz sind gegeben (Pirker/Berger/Watzke 2004) und werden in Abschnitt 3.3.3 beschrieben.

Software-Agenten eignen sich zur Sammlung, Filterung (Mabry et al. 2003, 260) und Aufbereitung von Informationen aus heterogenen und verteilten Datenquellen (Krcmar/Horn 2001a, 4). Dazu können Agenten an den Speicherort der Daten migrieren (Mabry et al. 2003, 259) und diese aus den Informationsquellen extrahieren. Nach der Aufbereitung können die Agenten an ihren ursprünglichen Ort zurückkehren. Damit bilden Software-Agenten eine adäquate Basis für die Umsetzung der im Gesundheitswesen gewünschten Informationsintegration. Vorarbeiten zur Realisierung dieser Anforderungen sind bei *Kamel Boulos et al.* (2006) gegeben, wobei der Fokus auf der Datenaggregation unter Berücksichtigung von Datenschutzeigenschaften liegt. Mit diesem Ansatz besteht insbesondere die Möglichkeit, die in Abschnitt 1.1 geforderte Möglichkeit der Steuerung des Gesundheitswesens zu realisieren, die durch die Bereitstellung von anonymisierten Datensätzen unterstützt werden kann.

Auch auf der Ebene von Informationssystemen besteht die Notwendigkeit, einen Paradigmenwechsel vorzunehmen: Der Bedarf einer intentionalen und Domänen unabhängigen Modellierung von Informationssystemen wird bei *Lyyntinen/Lehtinen* (1984) und *Schoop* (1998a) beschrieben (siehe dazu auch Abschnitt 3.5). Dieser Ansatz wird bei *Schoop* (1999; 1999; 2001) auf die Domäne Gesundheitswesen übertragen. Diese intentionale Modellierung von Informationssystemen, die nach *Schoop* (1999; 1999) insbesondere im Gesundheitswesen erforderlich ist, wird durch agentenbasierte Systeme vorteilhaft unterstützt; denn in einem solchen System mit einer Sprechakt basierten Kommunikation lässt sich die intentionale Modellierung geeignet vornehmen.

Nach der Abbildung eines CAS auf ein MAS bzw. der Begründung der Implementierung eines Informationssystems auf der Basis von Software-Agenten ist als nächster Schritt ein geeigneter Agentenbegriff auszuwählen. Aus der in Abschnitt 2.3.1 dargestellten Klassifikation von Agenten wird für die folgenden Ausführungen der minimale Agentenbegriff (siehe dazu Abschnitt 2.3.1.1) zugrunde gelegt. Dieser Terminus ist geeignet, weil sich mit den Eigenschaften eines minimalen Agenten die wesentlichen Elemente eines CAS modellieren lassen:

Der minimale Agentenbegriff zeichnet sich durch die Eigenschaften Autonomie, soziale Fähigkeit, Reaktivität und Proaktivität (Wooldridge/Jennings 1995) aus. Autonome Agenten sind in der Lage, nicht notwendigerweise antizipierbares Verhalten zu zeigen und eigenständig zu handeln. Die Individualität manifestiert sich insbesondere auch in der Proaktivität von Agenten, denn damit können Ziele von Agenten eigenständig angestrebt werden. Die soziale

Fähigkeit erlaubt Agenten, Interaktionen mit anderen Agenten zu etablieren. Diese Interaktionen können zu Aktionen führen, die den Kontext von Agenten verändern. Diese Veränderung kann von reaktiven Agenten wahrgenommen werden, um darauf geeignet und flexibel reagieren zu können. Eine Reaktion kann z.B. die Bildung eines Subsystems von Agenten sein, das gemeinsam an einer Problemlösung arbeitet. Insgesamt lässt sich festhalten, dass Interaktionen, Dynamik und Adaptivität als wesentliche Eigenschaften des als CAS modellierten Gesundheitswesens durch den minimalen Agentenbegriff abgebildet werden können. Die hohe Anzahl von Agenten, die jeweils Akteure des Gesundheitswesens repräsentieren, ist in einem MAS per se gegeben.

Entsprechend der Darstellung in Abschnitt 2.3.1.5 können Software-Agenten mit unterschiedlichen, internen Architekturen implementiert werden. In der vorliegenden Arbeit wird die Realisierung in Anlehnung an die Objektorientierung gewählt. Diese interne Architektur ist angesichts der angestrebten Konstruktion eines agentenbasierten Informationssystems für das Gesundheitswesen ausreichend, denn dadurch sind die Voraussetzungen für die geforderte Flexibilität, implementiert durch eine geeignete Modularisierung sowie lose Kopplung, gegeben.

Zusammenfassend lässt sich feststellen, dass sich das Gesundheitswesen geeignet durch ein CAS modellieren lässt. Letzteres kann vorteilhaft auf ein Agentensystem abgebildet werden, wobei der minimale Agentenbegriff sowie eine interne Architektur von Agenten in Anlehnung an die Objektorientierung ausreichend sind. Somit lassen sich Informationssysteme im Gesundheitswesen vorteilhaft auf der Basis von Software-Agenten konstruieren (Mouratidis et al. 2002).

Im folgenden Abschnitt 5.3.2 wird eine für ein agentenbasiertes Informationssystem geeignete Architektur abgeleitet.

5.3.2 Wahl einer geeigneten Agentenarchitektur

Aus der Eignung zur Abbildung von Eigenschaften des Gesundheitswesens auf Agenten folgt die Notwendigkeit zur Wahl einer adäquaten Architektur für ein MAS. Prinzipielle Überlegungen zu etablierten Architekturen wurden bereits in Abschnitt 3 angestellt. Aufbauend auf diesen Ausführungen werden im Folgenden die für eine Implementierung eines zukünftigen Informationssystems im Gesundheitswesen relevanten Architekturstile identifiziert.

Grundsätzlich ist das allgemeine Architekturprinzip der Schichtenbildung (siehe Abschnitt 3.2.3.1) auch für ein Informationssystem im Gesundheitswesen anzuwenden, um die Komplexität zu reduzieren und die Weiterentwicklungsfähigkeit zu unterstützen. In der vorliegenden Arbeit ist das Agentensystem deshalb geeignet in eine logische Schichtenarchitektur hineinzuentwickeln.

Das Gesundheitswesen ist per se ein verteiltes System, dementsprechend besteht für ein geeignetes Informationssystem die Anforderung nach einem verteilten System (siehe Abschnitt 3.2.3.2). Dabei kann zwischen Client-Server-Architekturen (siehe Abschnitt 3.2.3.3) und Peer-to-Peer-Ansätzen (siehe Abschnitt 3.2.3.4) differenziert werden, wobei mit der flexiblen Peer-to-Peer-Architektur die wesentlichen Einschränkungen eines Client-Server-Systems beseitigt werden können. Zu bemerken sind hier insbesondere die Skalierbarkeit, Sicherheit und Verfügbarkeit sowie die Flexibilität und Dienstgüte bei der Integration neuer Dienste (Stein-

metz/Wehrle 2004, 51). Da diese Eigenschaften für den Fokus der vorliegenden Arbeit nicht relevant sind, wird auf die Implementierung eines agentenbasierten Systems auf der Basis einer Peer-to-Peer-Architektur verzichtet. Die dafür erforderlichen und bei *Pirker/Berger/Watzke* (2004) beschriebenen Mechanismen sind aber so gestaltet, dass eine Integration in eine FIPA kompatible Agentenplattform und damit in die Architektur der vorliegenden Arbeit gewährleistet ist.

Nach *Lockemann/Nimis* (2005, 30) kann ein Agent auch im Sinne einer Komponente (siehe auch Abschnitt 3.3.5) betrachtet werden, die einen bestimmten Dienst anbietet. Dieser Zusammenhang kann auch auf abstrakterer Ebene betrachtet werden, indem mehrere Agenten eine Gruppe (siehe dazu auch Odell/Nodine/Levy 2004) bilden und somit als Gesamtheit einen Dienst implementieren. Mit dieser Zusammenstellung können Agenten, die an sich eine fein granulare Einheit darstellen, nach Bedarf zu komplexeren Komponenten integriert werden und damit einen wesentlichen Beitrag dafür leisten, die Komplexität zu reduzieren (siehe dazu auch Jennings 2001) und flexibel auf sich ändernde Anforderungen zu reagieren. Mit diesem Ansatz lässt sich z.B. das Prinzip der Informationslogistik vorteilhaft abbilden, indem eine Gruppe von Agenten dazu beauftragt wird, die benötigten Informationen zur Verfügung zu stellen. Dieser Ansatz skaliert dabei, weil entsprechend den Aufgaben eine geeignete Anzahl von Agenten eingesetzt werden kann.

Die Integration von Informationssystemen, wie sie für die integrierte Versorgung angestrebt wird, kann auf Basis der Konzepte der EAI (siehe Abschnitt 3.2.4.1) und SOA (siehe Abschnitt 3.2.4.2) vorteilhaft durch ein Agentensystem unterstützt werden (siehe dazu Abschnitte 3.3.6 und 3.3.7). Dabei erweist sich die besonders ausgeprägte Flexibilität in einem Agentensystem, induziert durch eine Modularisierung und lose Kopplung, als besonders vorteilhaft.

Mit diesen Ausführungen wird die Anforderung erhoben, die folgenden Architekturprinzipien in dem zu konstruierenden Agentensystem umzusetzen:

- *Schichtenarchitektur*
- *Flexible Komposition und Dekomposition von Agentensubsystemen*
- *EAI*
- *SOA*

Die Evaluierung von Software-Architekturen in Abschnitt 3.4.2 zeigt, dass die Flexibilität bei agentenbasierten Architekturen am stärksten ausgeprägt ist. Die Konzepte der weiter oben identifizierten, relevanten Architekturen werden deshalb in der vorliegenden Arbeit agentenbasiert umgesetzt.

5.4 Realisierung des Anforderungskatalogs über ein agentenbasiertes System

Die bisherigen Ausführungen in Abschnitt 5 zeigen, dass sich Anforderungen im Gesundheitswesen geeignet durch ein Agentensystem abbilden lassen. Zu zeigen ist nun eine entsprechende Umsetzung der Kriterien aus dem in Abschnitt 4.12 dargestellten Anforderungskatalog.

Zur Integration in eine verteilte Kommunikationsplattform (Anforderung 1 aus Abschnitt 4.12.1) eignet sich ein Agentensystem, weil es per se ein verteiltes System darstellt. Damit ist

es in der Lage, auf der Basis der TI für die eGK Mehrwertdienste anzubieten. Als Implementierungsplattform ist eine geeignete Laufzeitumgebung als Basis einer Agentenplattform auszuwählen, die eine Integration heterogener Informationssysteme und Endgeräte erlaubt.

Anforderung 2 aus Abschnitt 4.12.3 (Unterstützung von syntaktischen Standards) kann insofern über ein agentenbasiertes System erfüllt werden, als Agenten im Sinne von Informations-Brokern (Schweiger et al. 2006b) eingesetzt werden können und damit zwischen unterschiedlichen Standards vermitteln können, indem auf der Basis einer gemeinsamen Ontologie Informationen ausgetauscht werden. Im Zusammenhang mit den Ausführungen zur EAI bzw. SOA können damit Funktionalitäten des Message Brokers bzw. Dienstbus umgesetzt werden. Dabei können insbesondere auch etablierte Standards aus der medizinischen Domäne berücksichtigt und geeignet im Rahmen einer semantisch reichhaltigen Nachricht zwischen Agenten ausgetauscht werden. Damit kann auch Anforderung 4 aus Abschnitt 4.12.6 zur Unterstützung von semantischen Standards geeignet erfüllt werden.

Voraussetzung für eine lebenslange virtuelle Patientenakte ist die Existenz einer eindeutigen Patientenidentifikation, die mit der eGK zur Verfügung gestellt wird (Neuhaus/Deiters/Wiedeler 2006, 334), oder eine geeignete Verknüpfung Institutionen abhängiger Identifikationen (Wang/Ohe 1999). Eine eindeutige Identifikation aus Anforderung 3 (Abschnitt 4.12.4) ist mit der Einführung der eGK gegeben. Deshalb wird im Folgenden diese Alternative favorisiert. Für einen Patienten ist ein adäquates Konstrukt auf Software-Ebene erforderlich, welches entsprechend seiner Patientenidentifikation die zugehörigen Dokumente aggregiert. Dieser Mechanismus wird durch die Metapher der aktiven medizinischen Dokumente (AMD) realisiert, welche ausführlich in Abschnitt 6.5.1 beschrieben wird. Die grundsätzliche Idee ist die Kapselung von allen relevanten Daten für einen Patienten in einer Agentenstruktur. Somit wird die Kapselung auf reiner Attribut- und Methodenebene aus der Objektorientierung (siehe Abschnitt 2.2.5) auf höherer Abstraktionsebene auf semantische Inhalte im Sinne von Geschäftsobjekten übertragen, wie sie auch mit dem EAI-bzw. SOA-Ansatz (siehe auch Richter/Haller/Schrey 2005, 413) angestrebt wird.

Die Anwendungssäulen der Informationslogistik, Entscheidungs- und Prozessunterstützung aus Anforderung 5 in Abschnitt 4.12.7 können, wie in Abschnitt 5.5.2 beschrieben wird, geeignet in einem Agentensystem umgesetzt werden. Beispielhaft wird die in dieser Arbeit beschriebene Implementierung (siehe Abschnitt 6) die Informationsintegration aus heterogenen Informationssystemen, die Prozessunterstützung mit der Beschreibung und Abarbeitung eines Behandlungspfades sowie die Entscheidungsunterstützung an einem Beispiel aus der medizinischen Domäne demonstrieren. Mit dieser Funktionalität wird insbesondere der aktive Teil eines AMDs determiniert. Weiterhin wird damit eine Prozessintegration im Sinne der EAI bzw. SOA erreicht. Um der Vielfalt der unterschiedlichen Anwendungen gerecht zu werden, sind geeignete Architekturprinzipien zu berücksichtigen.

Aus dem Anforderungsbereich 6 (Abschnitt 4.12.8) können Agentensysteme insbesondere zur Implementierung eines flexiblen Informationssystems eingesetzt werden. Dabei können Software-Agenten ihre wesentlichen Vorteile gegenüber anderen Anwendungen besonders demonstrieren. Deshalb wird dieser Bereich aus dem Anforderungskriterium 6 ausgewählt. Das vorgeschlagene AMD-Konzept erlaubt dabei z.B. die flexible Delegation von Aggregations-

aufträgen an dedizierte Agenten oder dynamisches Scheduling. Die Patientenorientierung manifestiert sich u.a. in der Bereitstellung von Informationen für den Patienten. Weil dazu bereits Ansätze existieren (siehe dazu Abschnitte 4.9 und 4.9.2), ist in der prototypischen Implementierung in der vorliegenden Arbeit lediglich zu demonstrieren, dass sich eine Web basierte Patientenplattform in ein Agentensystem integrieren lässt und sich so die grundlegenden Funktionalitäten eines eKiosks im Sinne der TI realisieren lassen. Dazu sollte die Lösung auf etablierten Standards basieren, welche auch im Hinblick auf den zunehmenden Einsatz von Service orientierten Architekturen relevant sind, also z.B. auf Web Services.

Die Ausführungen in diesem Abschnitt zeigen, dass Agentensysteme den Anforderungskatalog geeignet realisieren können. Weiterhin zeichnen sich agentenbasierte Systeme insbesondere durch ihre ausgeprägte Flexibilität aus, weshalb sie für ein Integrationsszenario prädestiniert sind. Die durch ein agentenbasiertes System umzusetzenden Anforderungen aus diesem Abschnitt sind in der vorliegenden Arbeit anhand eines ausgewählten Anwendungsfalles weiter zu konkretisieren (siehe Abschnitt 6.3.6).

Das Gesundheitswesen wird bei Jennings/Wooldridge (1998, 15-16) als ein geeignetes Anwendungsgebiet für Agentensysteme explizit genannt. Folglich können mehrere Ansätze identifiziert werden, die Agentensysteme im Gesundheitswesen einsetzen. Ein Überblick über ausgewählte Arbeiten wird im folgenden Abschnitt 5.5 beschrieben.

5.5 Verwandte Arbeiten

5.5.1 Agent.Hospital

Mehrere heterogene und agentenbasierte Systeme werden im Agent.Hospital-Rahmenwerk (Paulussen et al. 2003a; Kirn et al. 2006a) zu einem Gesamtsystem integriert. Um diese MASe zu verknüpfen, werden Gateway-Agenten eingesetzt (Krempels et al. 2003, 38). Auf Implementierungsebene werden dazu die etablierten Entwurfsmuster Facade (Gamma et al. 1995, 185-193) und Singleton (Gamma et al. 1995, 127-134) angewendet (Krempels et al. 2003, 38).

Die Steuerung dieses Systems erfolgt über ein zentrales Simulationssystem (Heine/Herrler/Kirn 2005), welches den Kontrollfluss zwischen den Agentensystemen realisiert. Erhält ein Agentensystem den Kontrollfluss, wird die Weiterverarbeitung dort vorgenommen, bis die Kontrolle wieder zurück an das zentrale Simulationssystem delegiert wird.

Auf Anwendungsebene werden mit den heterogenen Agentensystemen unterschiedliche Einsatzbereiche innerhalb eines Krankenhauses abgedeckt. Die Teilsysteme umfassen dabei die folgenden Funktionalitäten (Kirn et al. 2006a, 203-205):

- *Policy Agents*: Ziel dieser Arbeiten (Czap/Becker 2002, 2003) ist die automatische Planung für OP-Säle. Eine besondere Herausforderung besteht in der Berücksichtigung der Interessen der involvierten Personen und Abteilungen. Dazu repräsentiert ein Agent jeweils einen Akteur mit einem spezifischen Präferenzprofil. Mit diesem Ansatz können Terminpläne gefunden werden, deren Erstellung solche Kosten verursacht, die deutlich unter den sonst üblichen Transaktionskosten liegen.
- *MedPAge*: Angestrebt wird mit diesem Ansatz (Awizen/Paulussen 2001; Paulussen et al. 2003b) eine Funktionsbereiche übergreifende dezentrale Planung, Steuerung und Koordination von klinischen Prozessen. Dazu repräsentieren Agenten Ressourcen und Pa-

tienten mit Präferenzen. Die Patientenagenten verhandeln untereinander um die erforderlichen Ressourcen. Das Verhandlungsmodell basiert auf einem Marktmechanismus, in dem die Verhandlungen so lange fortgeführt werden, bis ein Zustand mit der Eigenschaft Pareto-Effizienz (detaillierte Ausführungen dazu werden in Abschnitt 6.7.4.3 dargestellt) erreicht werden kann.

- *EMIKA*: Fokussiert wird in diesen Arbeiten (Sackmann/Eymann/Müller 2002, 12; Niemann/Eymann 2006) die Realisierung einer Echtzeitkoordination der Patientenlogistik in der Radiologie. In dem dezentralen Ansatz repräsentieren Software-Agenten die Elemente der realen Welt, welche z.B. über RFID-Chips wahrgenommen werden können. Diese Modellierung wird mit vorgesehenen Terminplänen verglichen, um ggf. bei Abweichungen autonom eine erneute Planung vorzunehmen. Angestrebt wird dabei die Integration von Notfällen in die bisherigen Planungen möglichst ohne Verzögerung.
- *ADAPT*: Ziel des Ansatzes (Herrler/Heine 2004; Heine/Herrler/Kirn 2005) ist die Unterstützung der Planung von klinischen Studien. Dazu werden relevante Prozesse für klinische Studien über ein Agentensystem simuliert (Heine et al. 2003), um die Analyse, Evaluierung und die Planung zu vereinfachen (Herrler/Heine/Klügl 2002).
- *ASAinlog*: Im Zentrum steht bei diesen Arbeiten[30] die Informationslogistik für eine elektronische Patientenakte. Dabei wird einerseits die Bereitstellung von Informationen nach dem informationslogistischen Prinzip fokussiert. Andererseits wird die Unterstützung für Prozesse der Koordination, Kommunikation und Kooperation des medizinischen und pflegerischen Personals angestrebt. Realisiert werden diese Arbeiten über das Konzept der aktiven medizinischen Dokumente auf der Basis von Software-Agenten (siehe Abschnitt 6.5.1 für Details). Die in diesem Teilprojekt erzielten Ergebnisse sind Grundlage für die in der vorliegenden Arbeit beschriebene Lösung.
- *AGIL²*: Der Fokus liegt auf einer durchgängigen Entwicklung eines Agentensystems (Rose et al. 2006). Dazu werden zusammen mit Domänenexperten relevante Prozesse modelliert, für die Identifikation von potenziellen Einsatzszenarios für Agenten analysiert und schließlich durch den Einsatz von Agenten optimiert. Dabei führen letztendlich Agenten Aufgaben aus, die vor der Einführung des entwickelten Agentensystems manuell ausgeführt werden mussten. Damit wird eine Steigerung der Effizienz erzielt.

Die beschriebenen Arbeiten werden entlang einem ausgewählten Behandlungsprozess integriert (Kirn et al. 2006a, 212-216). Dieser erstreckt sich jedoch nur über eine einzige Institution. Folglich werden hier Anforderungen der TI für die eGK und insbesondere Aspekte zur Institutionen übergreifenden Informationsintegration nicht berücksichtigt. Außerdem wird dabei die Integration von etablierten Informationssystemen vermisst, die jedoch für das Ziel der integrierten Versorgung relevant ist.

Zusammenfassend lässt sich feststellen, dass die genannten Ansätze wesentliche Ergebnisse für die Entwicklung von Agentensystemen und ihre Applikation in der Domäne Gesundheitswesen leisten. Dennoch lassen sich Defizite identifizieren, die mit der vorliegenden Arbeit beseitigt werden sollen.

5.5.2 Weitere Beispiele für Agentensysteme im Gesundheitswesen

Der Einsatz von Agentensystemen kann neben den in Abschnitt 5.5.1 beschriebenen Anwendungsmöglichkeiten zur Informationslogistik und Prozessunterstützung in weiteren Arbeiten insbesondere zur Informationslogistik, und -integration sowie zur persönlichen Assistenz bzw. für den Bereich ambiente Intelligenz beobachtet werden. Arbeiten dazu werden im Folgenden überblicksartig beschrieben:

[30] Auf Arbeiten, die aus diesem Projekt hervorgingen, wird in Abschnitt 1.12 verwiesen.

Agentensysteme eignen sich für den Einsatz in verteilten und heterogenen Umgebungen, in denen zusätzlich eine umfangreiche Informationsmenge zu erwarten ist (siehe auch Koutkias/Chouvarda/Maglaveras 2005, 528). In dem Anwendungsbeispiel eines Systems für die medizinische Betreuung chronisch Kranker (Koutkias/Chouvarda/Maglaveras 2005) werden Daten der Patienten, die z.B. die aktuelle Befindlichkeit beschreiben, von zu Hause in eine medizinische Einrichtung übertragen. Die übertragenen Datensätze werden durch ein Agentensystem auf Basis der FIPA-kompatiblen Plattform JADE (siehe dazu Abschnitt 6.3.2) überwacht und nach der Dringlichkeit für ihre Bearbeitung durch das medizinische Personal klassifiziert. Weiterhin werden unterschiedliche Sichten für das medizinische Personal und die Administration auf das generierte Datenmaterial ermöglicht.

Mit den Ergebnissen aus einer ersten Studie konnten *Koutkias/Chouvarda/Maglaveras* bereits Folgendes zeigen (2005, 536): Im Vergleich zu dem bisher eingesetzten herkömmlichen System konnte ein Vorteil hinsichtlich des Arbeitsaufwandes für das medizinische Personal erzielt werden, weil Patientendaten nach ihrer Dringlichkeit sortiert werden und damit die aufwändige Begutachtung jedes einzelnen Datensatzes zur Identifikation der Dringlichkeit des Falles entfällt. Mit dieser vom Agentensystem durchgeführten vorherigen Priorisierung der Datensätze konnten dringende Fälle umgehend bearbeitet und damit auch mehr Zeit und Aufwand in ihre Bearbeitung investiert werden. Insgesamt konnte in der vergleichenden Studie zwischen dem Einsatz eines herkömmlichen Systems und des agentenbasierten Systems festgestellt werden, dass der Arbeitsaufwand in dem dargestellten Szenario reduziert werden, damit Zeit eingespart und zusätzlich die Qualität der Behandlung gesteigert werden kann.

Die vorliegende Arbeit fokussiert statt der weiter oben beschriebenen Informationsfilterung mit der Informationsaggregation in einer virtuellen Patientenakte über mehrere Institutionen hinweg und unterscheidet sich in dieser Hinsicht von den Arbeiten bei *Koutkias/Chouvarda/Maglaveras* (2005).

Wiesman et al. (2006) beschreiben einen Ansatz zum agentenbasierten Information Retrieval mit der Integration von Suchmaschinen für medizinische Datenbanken wie z.B. MEDLINE (Wiesman et al. 2006, 29). Die Besonderheit dabei ist die Antizipierung des benötigten Wissens, um den Aufwand für die Informationssuche des medizinischen Personals zu reduzieren. In einer prototypischen Implementierung konnte die Tragfähigkeit des Konzepts demonstriert werden. Weiterhin ist die Integration einer elektronischen Patientenakte zu bemerken, die die Basis für die Aufbereitung der erforderlichen Informationen liefert. Weitere Details zu den Arbeiten von *Wiesman et al.* (2006) sind in Abschnitt 4.6.6 beschrieben.

Zusammenfassend kann festgestellt werden, dass mit dem Konzept von *Wiesman et al.* (2006) nur ein ausgewählter Problembereich in der medizinischen Domäne fokussiert wird. In dieser Hinsicht unterscheidet sich das in der vorliegenden Arbeit beschriebene Konzept einer modularen Architektur zur Abdeckung einer umfassenden Anzahl von Funktionalitäten.

Baujard et al. (1998) fokussieren in ihrer Arbeit die Bereitstellung von medizinischen Informationen aus dem Internet. Dabei ist in dem agentenbasierten Ansatz insbesondere die Unterstützung von mehreren Sprachen gegeben, wobei eine spezielle medizinische Suchmaschine eingesetzt wird (Baujard et al. 1998). Nicht berücksichtigt werden bei diesem Ansatz die In-

formationsintegration von Informationssystemen unterschiedlicher Institutionen und die Integration dieser Daten in einer elektronischen Patientenakte, wie sie in der vorliegenden Arbeit angestrebt werden.

Zur Aggregation von Informationen auf abstrakterer Ebene stellen *Kamel Boulos et al.* (2006) einen agentenbasierten Ansatz vor. Die wesentlichen Elemente davon sind in Abschnitt 2.3.10 zusammengefasst. Dabei liegt der Fokus auf der Wahrung der Vertraulichkeit der zu untersuchenden Daten und der Sicherheit. Das von *Kamel Boulos et al.* (2006) eingeführte Konzept erlaubt die anonymisierte Datenanalyse z.B. im Gesundheitswesen, um dieses geeignet steuern zu können (Roland Berger & Partner GmbH – International Management Consultants 1997, 10-11). Die Aggregation von Patienten bezogenen Daten wie in der vorliegenden Arbeit wird dabei aber nicht berücksichtigt.

Die Grundlage der Arbeiten von *Wiesman et al.* (2006) bzw. für einen geeigneten Einsatz des Vorschlags von *Kamel Boulos et al.* (2006) im Gesundheitswesen ist das Vorliegen einer elektronischen Patientenakte. Je mehr Informationen in dieser Akte enthalten sind, desto genauer können das Information Retrieval bzw. die Planung und Steuerung des Gesundheitswesens erfolgen. Deshalb liegt ein Fokus der vorliegenden Arbeit auf der Informationsintegration aus verteilten Informationssystemen in einer virtuellen Patientenakte im Sinne der Definition von *Roland Berger & Partner GmbH – International Management Consultants* (1997, 10 und 36-39). Damit kann eine ausreichende Informationsbasis zusammengestellt werden, um daraus weitergehende Arbeiten aufbauen zu können.

In den Arbeiten von *Zachewitz/Schwolow* (2004) bzw. *Zachewitz* (2004b; 2004a) werden Agenten zur Informationsintegration eingesetzt und können so als Grundlage für eine institutionsübergreifende Patientenakte fungieren (Zachewitz 2004a). Die Grundidee der Ergänzung der bisherigen Informationssysteme im Gesundheitswesen durch jeweils ein Agentensystem pro Institution sowie die Repräsentierung der Akteure durch geeignete Software-Agenten gleicht im Wesentlichen den Darstellungen in Abschnitt 6 bzw. 5.5.1. Die Konsistenz zwischen den aus den verteilten Informationssystemen aggregierten Daten wird über einen eigenen Maintenance-Mechanismus in einem siebenstufigen Phasenmodell erzielt (Zachewitz 2004a). Nicht berücksichtigt wird im Gegensatz zu der vorliegenden Arbeit die Integration von Potenzialen der TI. Weiterhin erstrecken sich die Arbeiten von *Zachewitz/Schwolow* (2004) bzw. *Zachewitz* (2004b; 2004a) im Gegensatz zu dem in dieser Arbeit vorgestellten Konzept nicht auf die Prozesssteuerung. Die Etablierung einer übergreifenden Patientenakte mit Patientenintegration in den Behandlungsprozess wird zudem nur angedeutet. Mit der Ergänzung der FIPA-Agentenplattform um eine Wartungskomponente, die bei *Zachewitz* (2004a; 2004b) beschrieben wird, können die Ergebnisse der Konsistenzsicherung sogar in das in dieser Arbeit beschriebene Konzept integriert werden. Insofern ergänzen diese Konzepte die vorliegende Arbeit.

Freßmann/Maximini/Sauer (2005) beschreiben einen agentenbasierten Ansatz zur Unterstützung von meist nicht vorhersagbaren und nicht wiederholbaren Prozessen (Freßmann/Maximini/Sauer 2005, 420), dessen Eigenschaften im Folgenden zusammengefasst werden (Freßmann/Maximini/Sauer 2005, 421-426 und 429):

Obwohl das Konzept primär im Zusammenhang mit der Durchführung von Feuerwehreinsätzen entwickelt wurde, kann dieses auch auf den medizinischen Bereich übertragen werden, weil darin ähnliche Anforderungen für die Flexibilität gelten (siehe dazu Abschnitt 4.2.2.2). In dem Ansatz werden insbesondere kontextsensitive Informationen berücksichtigt, um die Anwender bei der Ausführung der Prozesse adäquat zu unterstützen. Ausgangspunkt für die Prozessunterstützung zur Laufzeit ist die Auswahl eines geeigneten und abstrakten Prozesses. Die Prozessbeschreibung in Form einer Workflow-Definition wird mit einem Agenten assoziiert, der entweder den realen Akteur repräsentiert oder die in dem Workflow definierten Aufgaben ausführt. In der Komponente „Workflow Engine Manager" wird von der „Workflow Engine" die Instanz des abstrakten Workflows gebildet, um damit zu ermöglichen, dass diese entsprechend den sich dynamisch ändernden Rahmenbedingungen angepasst werden kann. Neben der Prozessbeschreibung enthält die „Workflow Engine" den aktuellen und lokalen Kontext. Relevante Daten fließen zur Laufzeit in einen globalen Kontextspeicher ein, der den anderen Teilnehmern des Systems zur Verfügung steht. Indem die Veränderungen des Kontextes und der Workflow-Definitionen registriert werden, wird eine Analyse von Abweichungen ermöglicht, die schließlich als eine Wissensbasis für zukünftige Prozesse dient. Der Einsatz eines agentenbasierten Systems ermöglicht die flexible Auswahl eines geeigneten Agenten, der eine Suchanfrage des Anwenders adäquat bearbeiten kann. Weiterhin erlauben Software-Agenten die Realisierung von Mustern für ihre Zusammenarbeit und Koordination. Schließlich sind Software-Agenten als Wrapper z.B. zu Suchmaschinen für die Anfragen der Anwender eingesetzt, um ein einheitliches Datenmodell zu realisieren. Damit erlauben Agenten die Integration unterschiedlicher Datenquellen.

Der Ansatz mit der provisorischen Auswahl eines geeigneten Prozesses und eine entsprechende Anpassung zur Laufzeit an die aktuellen Anforderungen ist prinzipiell für die Anwendung im Gesundheitswesen geeignet, eine konkrete Umsetzung dieser Idee ist bei *Freßmann/Maximini/Sauer* (2005) jedoch nicht beschrieben; denn vorausgesetzt wird dabei die Erstellung einer geeigneten domänenspezifischen Ontologie, auf deren Basis der Kontextbezug realisiert wird. Weiterhin ist eine direkte Integration der Prozessunterstützung in eine Patientenakte nicht umgesetzt, wie sie in der vorliegenden Arbeit angestrebt wird.

Über die Berücksichtigung der Informationslogistik hinaus gehen die Arbeiten von *Zachewitz* (2004b), sowie *Moreno/Isern/Sánchez* (2003) und *Moreno/Isern* (2002), indem persönliche Assistenten für Patienten im Sinne der ambienten Intelligenz (siehe Abschnitt 3.3) eingeführt werden. Neben der Assistenz des Patienten bei der Suche nach geeigneten medizinischen Institutionen werden eine übergreifende Patientenakte auf der Basis von Software-Agenten sowie eine Terminkoordination beschrieben. *Moreno/Isern/Sánchez* (2003) bzw. *Moreno/Isern* (2002) berücksichtigen in ihren Arbeiten jedoch nicht die Gegebenheiten des deutschen Gesundheitswesen sowie folglich auch nicht die Potenziale der TI für die eGK. Darüber hinaus ist bei *Moreno/Isern/Sánchez* (2003) keine Prozesssteuerung integriert und keine Portierung auf mobile Endgeräte gegeben. Insofern unterscheidet sich die vorliegende Arbeit von den genannten Ansätzen.

Die in Abschnitt 5.5.2 beschriebenen Arbeiten demonstrieren, dass der Einsatz von Software-Agenten im Gesundheitswesen ein viel versprechender Ansatz zur Lösung von informationslogistischen Problemstellungen sowie zur Unterstützung der persönlichen Assistenz im Sinne der ambienten Intelligenz ist. Dennoch fokussieren

die genannten Arbeiten jeweils nur spezifische Problemstellungen. Im Gegensatz dazu zielt das in der vorliegenden Arbeit dargestellte AMD-Konzept auf einen modularen Ansatz ab, der unterschiedliche Funktionalitäten zusammenführt bzw. eine Erweiterung explizit unterstützt.

5.6 Zusammenfassung

Ausgehend von der Modellierung des Gesundheitswesens als CAS wurde in Abschnitt 5 gezeigt, dass MASe vorteilhaft zur Abbildung eines CAS auf Elemente der Software-Technik eingesetzt werden können. Aus dieser transitiven Relation lässt sich ableiten, dass MASe Anforderungen für Informationssysteme im Gesundheitswesen erfüllen können. Weiterhin lässt sich feststellen, dass agentenbasierte Systeme die bisher fehlende intentionale Modellierung von Informationssystemen, insbesondere auch im Gesundheitswesen, vorteilhaft unterstützen. Aus der Vielzahl der Definitionen von Agenten wird für die vorliegende Arbeit der minimale Agentenbegriff ausgewählt. Dieser ist für die Unterstützung eines CAS auf der Software-Ebene geeignet.

Weiterhin wird demonstriert, dass sich Multiagentensystemarchitekturen dazu eignen, die Spezifika der Domäne Gesundheitswesen adäquat abzubilden. Dazu können insbesondere bewährte Konzepte von herkömmlichen Architekturen umgesetzt und erweitert werden. Dabei ist relevant, dass sich agentenbasierte Systeme hinsichtlich ihrer Flexibilität besonders auszeichnen. Der aus den in Abschnitt 4 gesammelten Anforderungskriterien für ein Informationssystem im Gesundheitswesen konzentrierte und in Abschnitt 4.12 beschriebene Katalog kann somit durch ein Agentensystem erfüllt werden.

Dass sich Agentensysteme über die in dieser Arbeit beschriebenen Konzepte hinaus vorteilhaft für Problemstellungen im Gesundheitswesen geeignet einsetzen lassen, demonstriert auch eine Vielzahl von wissenschaftlichen Arbeiten, die in einem Überblick in Abschnitt 5.5 beschrieben wurden. Die vorliegende Arbeit fokussiert die in diesen Arbeiten vermissten Funktionalitäten.

Mit den Ergebnissen aus diesem Abschnitt sind die Voraussetzungen für die Beschreibung des Designs und der Implementierung eines agentenbasierten Prototyps im folgenden Abschnitt 6 gegeben.

6 Anwendungsfall

6.1 Überblick

Um die in den bisherigen Ausführungen in dieser Arbeit beschriebenen, theoretischen Darstellungen in die Praxis zu übertragen, wird in Abschnitt 6 ein Anwendungsfall gewählt, anhand dessen ein Agentensystem konstruiert wird, das die Anforderungen aus dem in Abschnitt 4.12 erarbeiteten bzw. in Abschnitt 5.4 agentenbasiert fortgeführten Anforderungskatalog erfüllt. Die dafür erforderlichen Konzepte werden ausgehend von einem vollständigen Behandlungsprozess in einem Referenzklinikum in einem Trichtermodell bis zur Implementierung beschrieben.

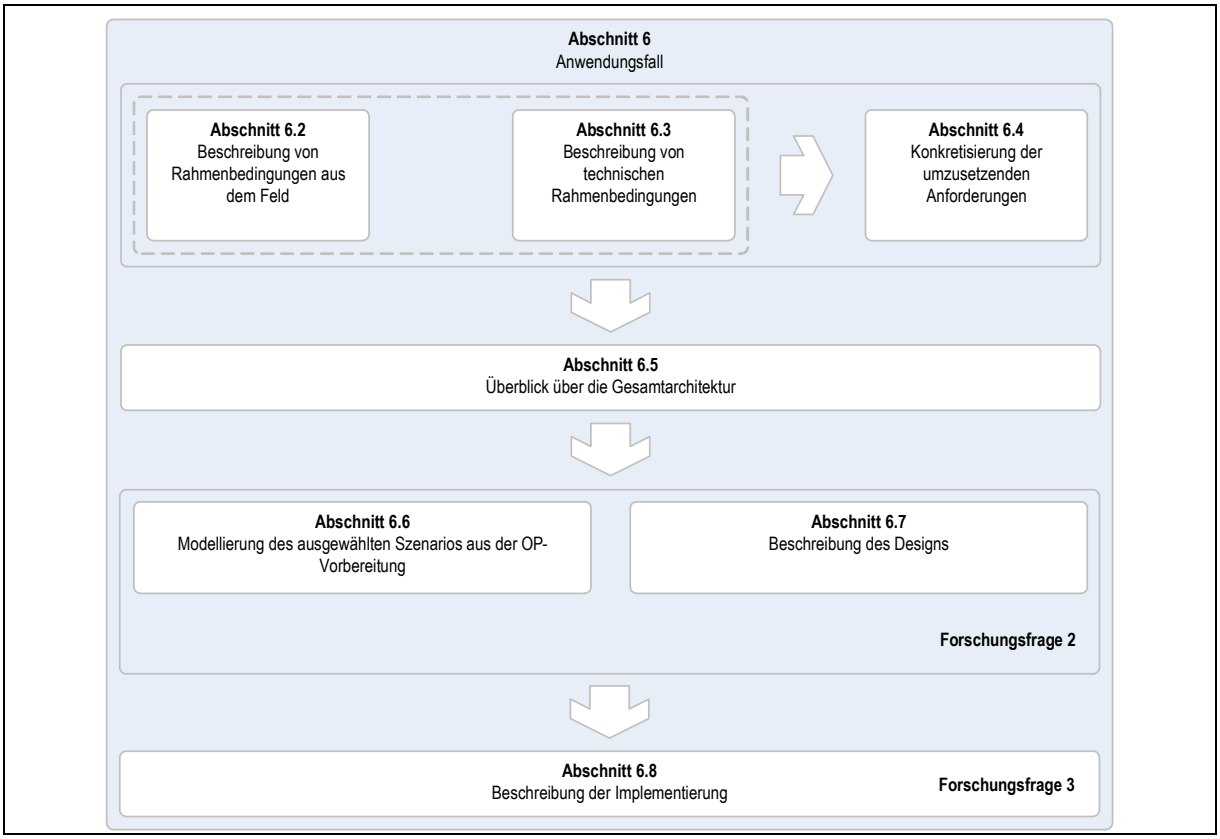

Abbildung 6.1-1: ***Überblick über Abschnitt 6***
Quelle: Eigene Darstellung

Dazu werden in Abschnitt 6.2 (Abbildung 6.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1) Rahmenbedingungen des gewählten Anwendungsfalles beschrieben. Diese werden ergänzt durch die Darstellung von technischen Eigenschaften der gewählten Implementierungsplattform in Abschnitt 6.3. Mit diesen Ausführungen kann der Anforderungskatalog in Abschnitt 6.3.6 hinsichtlich funktionaler und nicht funktionaler Eigenschaften weiter konkretisiert werden. Aus diesen Darstellungen wird in 6.5 eine geeignete Gesamtarchitektur abgeleitet. In diese wird das Design der gesamten Applikation hineinentwickelt (Abschnitte 6.6 und 6.7). Mit der Beschreibung des Designs ist Forschungsfrage 2 aus Abschnitt 1.4 zu beantworten. Die Darstellung des Designs wird durch die Beschreibung der

Implementierung in Abschnitt 6.8 konkretisiert. Damit ist Forschungsfrage 3 aus Abschnitt 1.4 zu beantworten.

6.2 Beschreibung von Rahmenbedingungen aus dem Feld

Die im Folgenden beschriebenen Ergebnisse resultieren aus der Anwendung des NDA (siehe dazu Abschnitt 1.6) in mehreren Fallstudien[31], die im gewählten Referenzklinikum durchgeführt wurden. Zur Reduktion der Komplexität wird in dieser Arbeit auf die Modellierung mit aktiven dynamischen Handlungsräumen (siehe dazu Abschnitt 1.6) als Erweiterung des NDA verzichtet.

Bei der im Folgenden beschriebenen Fallstudie zur Analyse des Behandlungsprozesses für das kolorektale Karzinom sind die folgenden NDA-Analyseaspekte relevant (Crosby 2005, 49):

- Organigramme (Modelltyp vorgeschlagen in Abschnitt 1.6) zur Darstellung der formalen Organisationsstruktur
- Modellierung von Interaktionen sowie von erfassten Kommunikationskanälen in einem Interaktionsnetz wie bei *Schwabe/Krcmar* (1996b)
- Erfassung von Prozessen durch erweiterte ereignisgesteuerte Prozessketten
- Ontologie- bzw. Datenmodellierung (siehe zur Datenmodellierung die Ausführungen in Abschnitt 6.6.3.4)

Die für die Anwendungsdomäne relevanten und komplexen Prozessbeschreibungen werden in erweiterten ereignisgesteuerten Prozessketten (Keller/Meinhardt 1994) modelliert, die eine Weiterentwicklung der ursprünglichen ereignisgesteuerten Prozessketten (EPK, Keller/Nüttgens/Scheer 1992) hinsichtlich der Ergänzung mit Prozessschnittstellen und hierarchischen Funktionen darstellen. Die Modellierung von Domänenprozessen im Gesundheitswesen auf der Grundlage von EPKn wurde bereits mehrfach erfolgreich eingesetzt[32]. Weiterhin zeigte sich die Evaluierung der erstellten Modelle in Zusammenarbeit mit Domänenexperten auf der Grundlage von ereignisgesteuerten Prozessketten als geeignet, um mögliche Modellfehler zu identifizieren. Aus diesen Gründen wird als Einstiegspunkt für die Anforderungserhebung die EPK-Modellierung auch für die vorliegende Arbeit eingesetzt. Für die werkzeugunterstützte Modellierung der erweiterten ereignisgesteuerten Prozessketten (Keller/Nüttgens/Scheer 1992; Keller/Meinhardt 1994) fällt die Wahl auf das Werkzeug ARIS Toolset der IDS Scheer AG[33]. Die darin erstellten Modelle können im XML-Format exportiert werden. Damit wird die Möglichkeit für eine spätere automatisierte Weiterverarbeitung der Prozessdefinitionen in dem Agentensystem ermöglicht (siehe dazu Abschnitte 6.7.6 und 6.8.7).

Datenmodelle werden in der vorliegenden Arbeit in UML-Notation dargestellt und durch Ontologien ergänzt, die für die Entwicklung des agentenbasierten Systems relevant sind. UML

---

[31] *Schweiger/Krcmar* (2005), *Crosby* (2005), *Bastian* (2005), *Ortmann et al.* (2007).

[32] *Paulussen et al.* (2003a, 75, 77 und 79), *Schweiger/Krcmar* (2005), *Kirn et al.* (2006a), *von Eiff/Ziegenbein* (2003).

[33] Weitere Informationen zu diesem Werkzeug können auf den folgenden Seiten abgerufen werden: http://www.ids-scheer.de, zugegriffen am 14.04.2006.

(Unified Modeling Language™, Object Management Group 2007c) wurde gewählt, weil es sich dabei um ein in der objekt- bzw. agentenorientierten Software-Entwicklung etabliertes Vorgehen handelt (siehe dazu Abschnitt 2.3.9) und es auch als Basis für die Modellierung in der ArBaCon-Methode (siehe Abschnitt 1.7) eingesetzt wird.

Die Grundlagen für die Erstellung der genannten Modelle werden mit den Methoden Beobachtung, teilstrukturierte Interviews (Bortz/Döring 2002, 238f.) und Dokumentenanalyse erarbeitet (Crosby 2005, 48). Mit diesen Modellen wird der Ausgangspunkt für den weiteren Prozess zur Konstruktion des Agentensystems gebildet (zum gewählten Vorgehensmodell siehe Abschnitt 1.7).

Die erwähnten Modelle werden für den gewählten Anwendungsfall in den folgenden Abschnitten 6.2.1 und 6.2.3 mit 6.2.5 beschrieben.

6.2.1 Organisationsmodell zur Darstellung des Referenzklinikums

Das Referenzklinikum Klinikum rechts der Isar der Technischen Universität München (MRI) integriert die medizinische Versorgung und den universitären Lehrbetrieb über die medizinische Fakultät der Technischen Universität München und fungiert als ein Krankenhaus der Maximalversorgung, in dem sämtliche Gebiete der modernen Medizin abgedeckt werden (Crosby 2005, 51). In 2003 wurden 398.104 Patienten in insgesamt 1.333 möglichen Betten (Klinikum rechts der Isar der Technischen Universität München 2003, 7) von 3.714 Mitarbeitern (Klinikum rechts der Isar der Technischen Universität München 2003, 272) versorgt.

Die Fachbereiche wie Chirurgie, Innere Medizin etc. bilden im Klinikum rechts der Isar nicht Abteilungen, sondern weitgehend selbstständige Kliniken (für die folgenden Ausführungen siehe auch Abbildung 6.2-1). Diese stehen jeweils unter eigener ärztlicher Leitung und sind voneinander unabhängig, auch in Bezug auf die Auswahl und Verwendung von IT. Die einzelnen Kliniken des MRI sind deshalb nicht administrativ, aber medizinisch beinahe so selbstständig wie einzelne Krankenhäuser. Insbesondere sind sie rechtlich nicht daran gebunden, für die medizinischen Funktionen wie Leistungsanforderung, medizinische Dokumentation und Befundübermittlung das gemeinsame Klinikinformationssystem zu nutzen. Jede Klinik hat vielmehr die Möglichkeit, eigene IT-Lösungen einzuführen.

Weitere Bestandteile des Klinikums sind die interdisziplinären bzw. krankheitsorientierten Zentren, zu denen jeweils mehrere Kliniken beitragen. Diese sind das Tumortherapiezentrum (TTZ), Brustzentrum, Gefäßzentrum, endokrine Zentrum sowie Mutter-Kind-Zentrum.

Im analysierten Prozess für das kolorektale Karzinom steht das TTZ im Mittelpunkt. Letzteres wird definiert als ein interdisziplinäres Zentrum, welches unterschiedliche Kompetenzen zur Diagnose und Therapie u.a. für das genannte Krankheitsbild aggregiert (Crosby 2005, 51). Dieses Zentrum wird von neun Kliniken und drei Instituten betrieben, in welchen jeweils die eigentlichen Therapien vorgenommen werden (Crosby 2005, 51). In seiner Ambulanz werden Diagnostik, Therapie und Nachsorge abgedeckt (Crosby 2005, 51). Das TTZ ist somit ein zentraler Punkt der Behandlung für das kolorektale Karzinom, von dem aus relevante Ent-

scheidungen für das weitere Vorgehen getroffen werden. Aus Organisationssicht wird das TTZ wie eine Arztpraxis verwaltet und setzt ein eigenes Praxisverwaltungssystem ein.

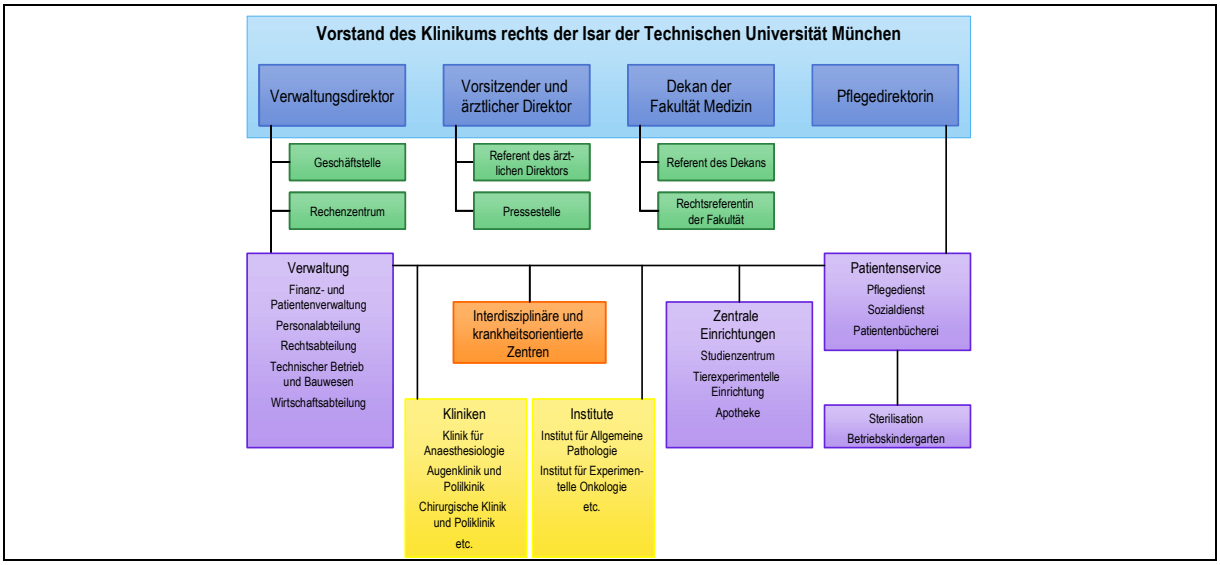

Abbildung 6.2-1: ***Organigramm des MRI***
Quelle: *Ortmann et al.* (2007, 6), vereinfacht nach *Klinikum rechts der Isar der Technischen Universität München* (2006)

6.2.2 Auswahl eines Behandlungsprozesses

Aus den im Folgenden dargestellten medizinischen und aufbauorganisatorischen Gründen wurde bewusst der Behandlungsprozess für das kolorektale Karzinom im Klinikum rechts der Isar der Technischen Universität München gewählt:

In einem Fünfjahreszeitraum wird in Deutschland bei 160.000 Menschen diese Krankheit diagnostiziert (Birkner 2003). Es handelt sich dabei außerdem um diejenige Krebserkrankung mit der zweithäufigsten Todesfolge (Crosby 2005, 54). Die genannte Erkrankung kann somit als eine der häufigsten Krebserkrankungen eingestuft werden, obwohl die Anwendung von existierenden Präventionsmaßnahmen zu reduzierter Inzidenz und Mortalität führen würde (Birkner 2003). Das Potenzial der Präventionsmaßnahmen wird bei *Birkner* (2003) als noch nicht ausgeschöpft eingeschätzt. Aus medizinischer Sicht kann der gewählte Behandlungsprozess damit als relevant betrachtet werden.

Aus Organisationsperspektive ist an dem genannten Prozess ein breites Spektrum an verschiedenen Disziplinen beteiligt, die auf eine Vielzahl von diagnostischen und therapeutischen Maßnahmen zurückgreifen. Außerdem handelt es sich bei dem Prozess um einen sektorenübergreifenden Ablauf, bei dem nicht nur das Klinikum MRI, sondern auch niedergelassene Ärzte involviert sind. Insbesondere hier zeigt sich der Bedarf der institutionsübergreifenden Informationsintegration und einer IT-Unterstützung zur Koordination im Behandlungsprozess. Auch eine Integration mehrerer Disziplinen innerhalb einer Institution impliziert die Anforderung der Kooperation und Koordination zwischen vielen Akteuren. Insbesondere ist dabei z.B. eine automatisierte und dynamische Terminplanung erwünscht. Die Anforderung um-

fasst dazu die Optimierung der Terminpläne von Einheiten mit diagnostischen und therapeutischen Einrichtungen und ggf. die Anpassung von Behandlungsprozessschritten. Dabei stehen Determinanten wie die Reduktion der Arbeitslast für die manuelle Terminkoordination, die Verkürzung der Wartezeiten für Patienten oder die Steigerung der Ressourcenauslastung im Vordergrund. Die Beteiligung mehrerer Disziplinen und Sektoren bedarf außerdem einer für das Anwendungsfeld angepassten Terminologie, um die semantische Integration zu gewährleisten.

Mit der Einführung eines geeigneten agentenbasierten Systems für einen ausgewählten Ausschnitt des genannten Behandlungsprozesses können die Verbesserung der Behandlungsqualität, -effizienz und damit auch eine Reduktion der Kosten für die eigentliche Behandlung oder der entstehenden Folgekosten angestrebt werden.

6.2.3 Interaktionsmodellierung

Da am betrachteten Prozess zur Behandlung des kolorektalen Karzinoms mehrere Disziplinen beteiligt sind, erhält die Koordination der involvierten Akteure einen wesentlichen Stellenwert. Ein Überblick über die beobachteten Interaktionen zwischen Akteuren bzw. den Organisationseinheiten sowie die jeweils verwendeten Medien werden, dem Vorgehen bei *Schwabe/Krcmar* (1996b) folgend, in Interaktionsnetzen dargestellt.

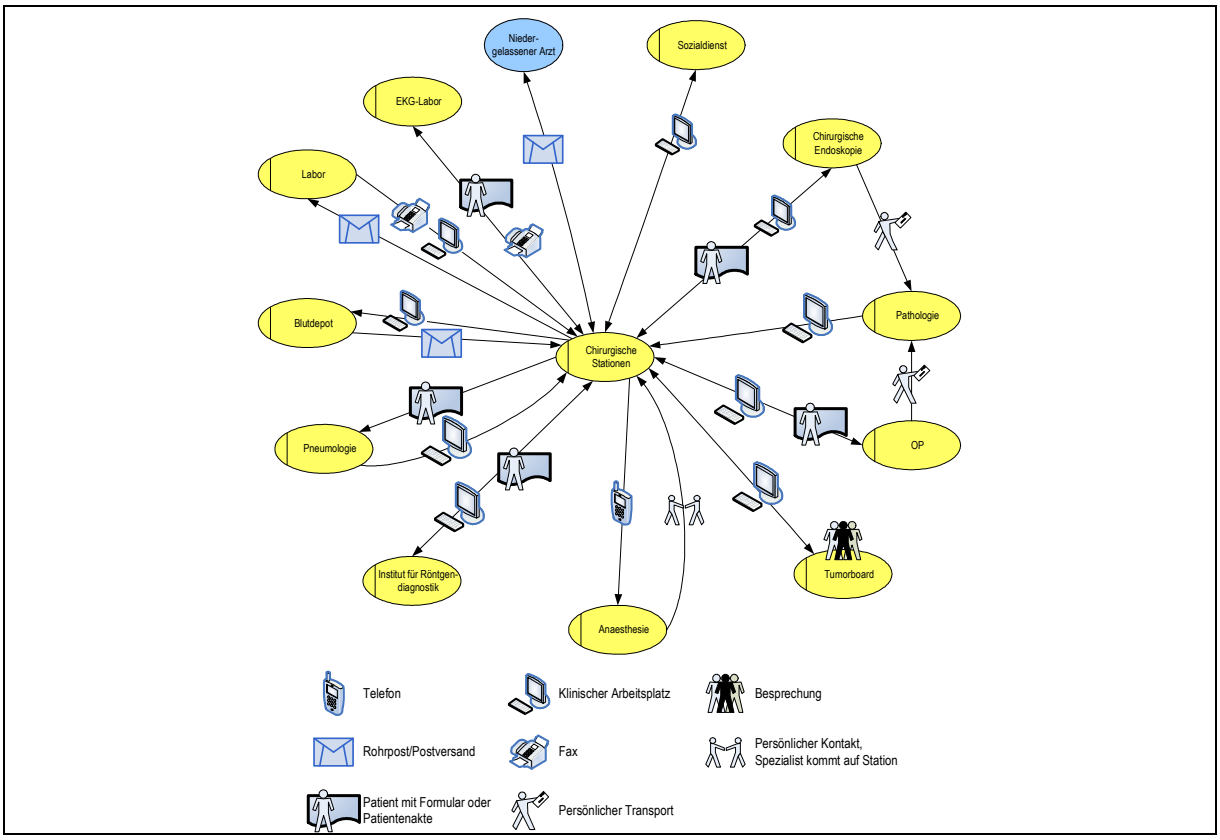

Abbildung 6.2-2: ***Interaktionen für die Teilprozesse Diagnostik und Operation***
Quelle: *Crosby* (2005, 82)

In Abbildung 6.2-2 werden für die Teilprozesse der Diagnostik und Operation (für die Beschreibung der gesamten Prozesssicht siehe Abschnitt 6.2.4) beispielhaft beteiligte Akteure und ihre Interaktionen dargestellt. Im Zentrum von Abbildung 6.2-2 befinden sich die chirurgischen Stationen (siehe auch Crosby 2005, 84), gemessen an der Anzahl der identifizierten Interaktionen. Hinsichtlich eingesetzter Kommunikationsmedien ist ersichtlich, dass für die Interaktionen nicht durchgängig IT-Lösungen eingesetzt werden. So wird z.B. die Kommunikation mit dem niedergelassenen Arzt durch Postversand realisiert. Weiterhin können bei den Interaktionen auch Medienbrüche beobachtet werden. Dies trifft z.B. auf die Kommunikation der chirurgischen Stationen mit dem Blutdepot oder dem Labor zu.

Abbildung 6.2-3 stellt exemplarische Interaktionsmuster für die Teilprozesse der Radio-Chemotherapie und der Nachsorge dar, wobei aus Gründen der Übersichtlichkeit nicht alle Kommunikationskanäle angegeben werden. Hier wird deutlich, dass das TTZ, gemessen an der Anzahl der Interaktionen, im betrachteten Behandlungsprozess eine zentrale Stellung einnimmt (siehe auch Crosby 2005, 84). Auch in diesem Interaktionsmodell können Medienbrüche und nicht IT-gestützte Kommunikation beobachtet werden. Um die Arbeitslast zu beherrschen, wird der Stationsarzt durch die Fachkräfte des TTZs und vom Pflegepersonal bei der Koordinationsarbeit unterstützt (Crosby 2005, 84). Damit wird deutlich, dass ein geeignetes IT-System zur Unterstützung der Koordination und zur Reduktion von Medienbrüchen einen Mehrwert schaffen würde, um die tägliche Arbeit des Personals zu vereinfachen.

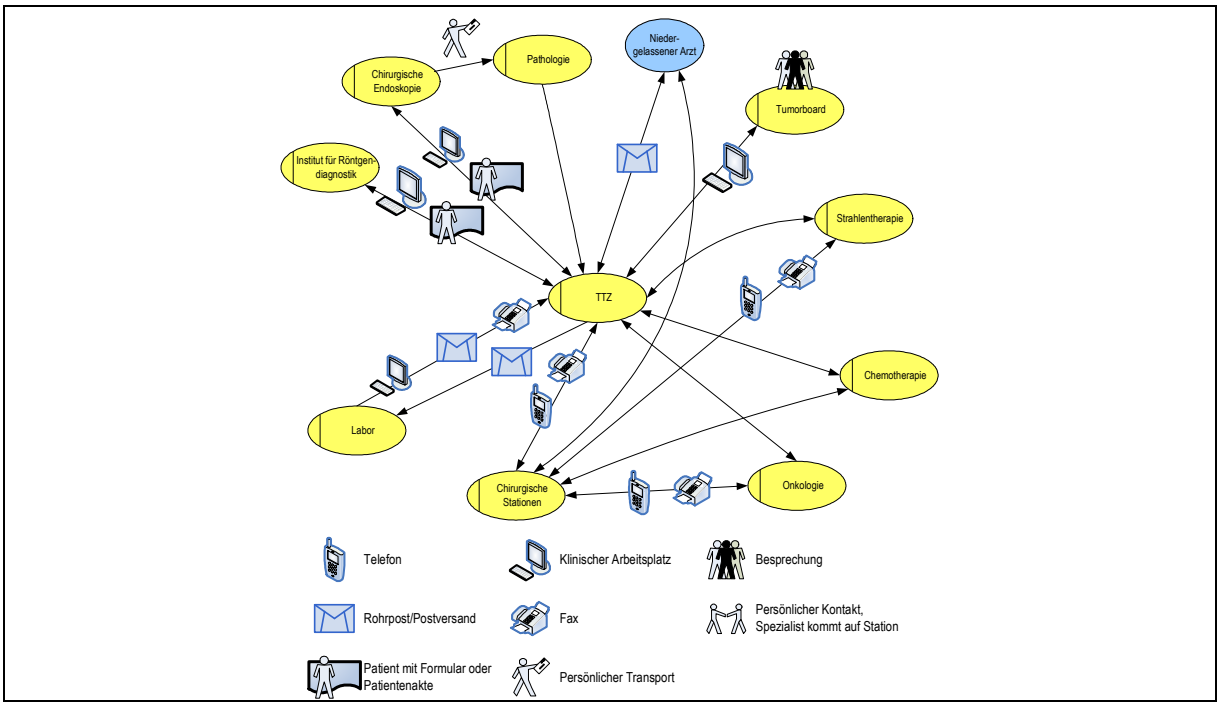

Abbildung 6.2-3: ***Interaktionen für die Teilprozesse Radio-Chemotherapie und Nachsorge***
Quelle: *Crosby* (2005, 83)

Insgesamt kann also festgehalten werden, dass eine Reihe unterschiedlicher Akteure aus selbstständigen Organisationseinheiten beteiligt ist, Medienbrüche bei der Kommunikation zwischen den Akteuren und nicht IT-unterstützte Kommunikation beobachtet werden können. Weiterhin ist zu bemerken, dass die eigentlichen Koordinationsaufgaben dem Stationsarzt ob-

liegen, die z.B. die Anforderung von Maßnahmen, Terminabsprachen, Dokumentation und Auswertung von Befunden umfassen (Crosby 2005, 84). Somit kann Bedarf für eine geeignete IT-Unterstützung identifiziert werden.

Um die Diversifizierung der eingesetzten Informationskanäle zu explizieren, werden diese in Abbildung 6.2-4 dargestellt. Dabei wird insbesondere ersichtlich, dass eine Reihe unterschiedlicher Medien eingesetzt wird, durch welche Medienbrüche induziert werden.

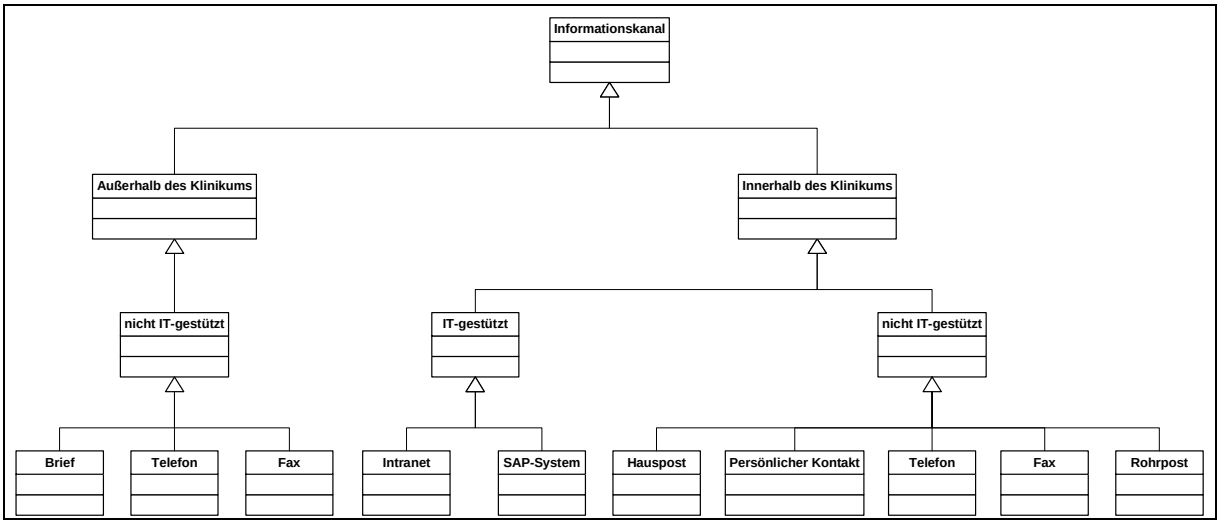

Abbildung 6.2-4: ***Informationskanäle im betrachteten Behandlungsprozess***
Quelle: In Anlehnung an *Crosby* (2005, 84)

Aus den dargestellten Interaktionsnetzen (siehe Abbildung 6.2-2 und Abbildung 6.2-3) wird ersichtlich, wie viele unterschiedliche Akteure in den jeweiligen Einrichtungen über heterogene Informationskanäle kommunizieren. Um diese Komplexität zu reduzieren, eignet sich in diesem Szenario die Konstruktion eines adäquaten Informationssystems, welches die jeweiligen Prozesse der Interaktion und Informationslogistik geeignet unterstützt.

6.2.4 Prozessmodellierung

Der Prozess zur Behandlung des kolorektalen Karzinoms wurde bei *Crosby* in einer ausführlichen Fallstudie (2005) analysiert und modelliert. Ersterer ist in Abbildung 6.2-5 (wegen der Übersichtlichkeit wird auf die bipartite Alternierung von Ereignissen und Funktionen verzichtet) in einer Grobübersicht dargestellt und wird im Folgenden beschrieben:

Wird beim Patienten durch den Haus- oder Facharzt (Gastroenterologe) ein kolorektales Karzinom diagnostiziert, erfolgt eine Überweisung in das Klinikum. Nach der Aufnahme im Klinikum werden im Teilprozess Diagnostik umfangreiche Untersuchungen durchgeführt. In einem Arztgespräch wird zunächst die Anamnese festgehalten. Es folgen zur Diagnostik Untersuchungen wie Labor, EKG, Lungenfunktion, Röntgen Thorax, Sonographie Abdomen, Endoskopie oder Computertomographie Abdomen. Ergänzt werden die diagnostischen Maßnahmen durch kardiologische, urologische bzw. gynäkologische Konsile. Im Tumorboard erfolgt die zentrale Entscheidung über das weitere Vorgehen in der Behandlung. Wird im Tumorboard auf Operation entschieden, erfolgen die Aufklärung des Patienten zur Anaesthesie sowie zur Operation, die eigentliche Operation, die Betreuung auf der Intensivstation und schließlich auf der Station. Entscheidet das Tumorboard auf Behandlung durch Radio-Che-

motherapie (RCTx), wird der Patient in der Onkologie bzw. im TTZ weiter behandelt. In Therapieeinheiten über mehrere Wochen erfolgt dann die Strahlen- bzw. Chemotherapie. Nach dem Abschluss der beschriebenen Teilprozesse OP bzw. RCTx muss über das weitere Vorgehen entschieden werden. Der Nachsorge kann eine Rehabilitation vorgelagert sein. Im Teilprozess der Nachsorge erfolgen Nachkontrollen in Abständen von drei bis sechs Monaten über den Zeitraum von fünf Jahren nach der OP. Werden innerhalb dieses Zeitraumes Rezidive oder Metastasen festgestellt, wird der Patient im Tumorboard erneut vorgestellt und der Behandlungsprozess wird wieder durchlaufen. Werden innerhalb der nächsten fünf Jahre keine Rezidive oder Metastasen identifiziert, gilt die Behandlung als erfolgreich abgeschlossen.

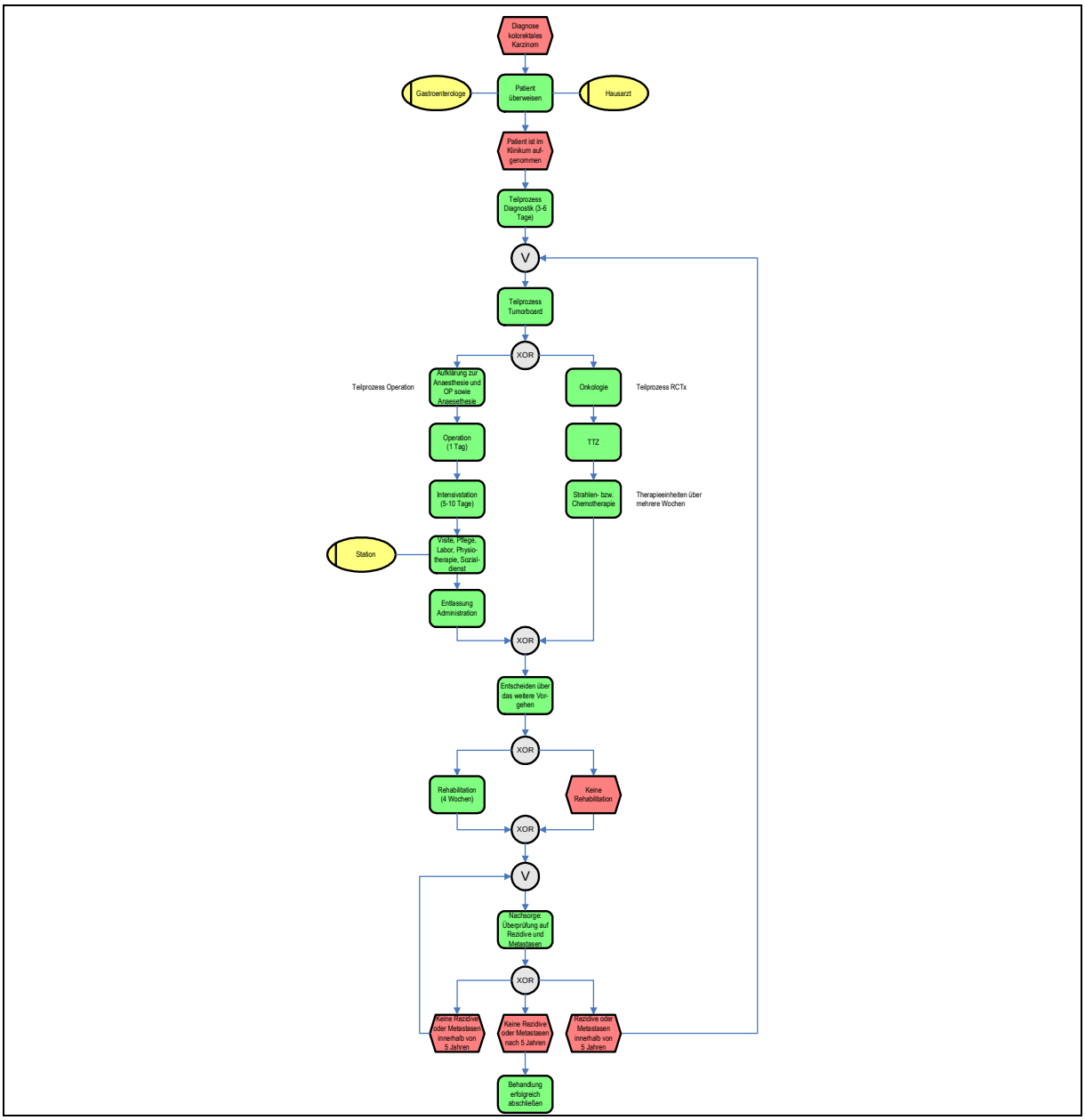

Abbildung 6.2-5: ***Kurzübersicht zum Behandlungsprozess kolorektales Karzinom am MRI***
Quelle: Eigene Darstellung, in Anlehnung an *Crosby* (2005, 115)

Diese Darstellung des Prozesses ist für das weitere Verständnis der vorliegenden Arbeit ausreichend, denn es erfolgt eine Konzentration auf ausgewählte Ausschnitte aus dem Behandlungsprozess, deren IT-Unterstützung schrittweise verfeinert und schließlich prototypisch implementiert wird. Die detaillierten Teilprozesse für die beschriebene Behandlung des kolorektalen Karzinoms sind ausführlich bei *Crosby* (2005, 55-84 und 116-131) dargestellt.

6.2.5 Modellierung der Ontologie

Eine gemeinsame Ontologie ist die Voraussetzung zur Informationsintegration aus unterschiedlichen Informationssystemen und damit zur Kommunikation von Informationen zwischen Agenten in einem Agentensystem. Die Nachrichten, welche zwischen Agenten ausgetauscht werden, basieren auf den in der Ontologie definierten Konzepten. Damit wird eine semantisch reichhaltige Kommunikation bei gleichzeitiger loser Kopplung des verteilten Systems erzielt.

Grundlage für die in der vorliegenden Arbeit eingesetzte Ontologie ist die für *Agent.Hospital* (Kirn et al. 2006a) entwickelte Ontologie *OntHoS* (siehe Becker et al. 2003; Kirn et al. 2006a, 205-207). Eine Beschreibung zu den in OntHoS (siehe Abbildung 6.2-6) definierten Konzepten ist in Tabelle 6.2-1 zusammengefasst. Dabei ist insbesondere das Bewertungskonzept ausführlich dargestellt, in dem zwischen absoluter und relativer Bewertung differenziert wird. Zu bemerken ist in diesem Zusammenhang insbesondere, dass die Bewertung nach dem deutschen Schulnotensystem die Abstufungen zwischen 1 und 6 enthält. Die Klassifizierung nach dem russischen Schulnotensystem erlaubt die Bewertung zwischen 5 und 1, wobei 5 die beste und 1 die schlechteste Note ist. Solche Bewertungen werden insbesondere für Terminvereinbaren benötigt.

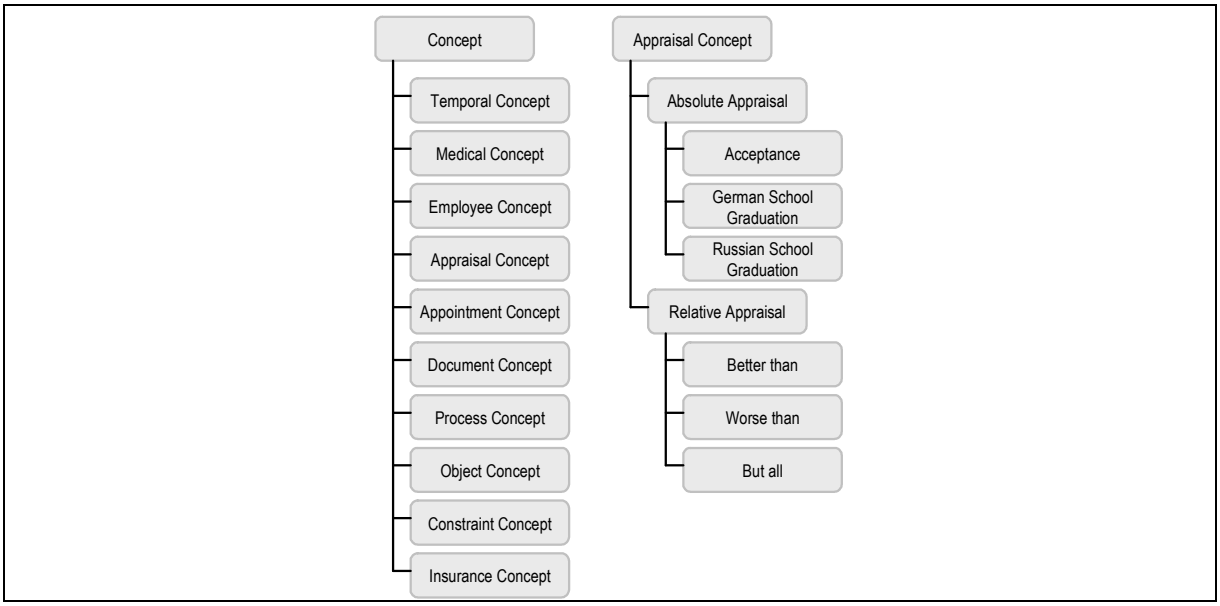

Abbildung 6.2-6: ***Ausgewählte Konzepte der Ontologie OntHoS***
Quelle: *Kirn et al.* (2006a, 206)

Um die Gegebenheiten des Referenzklinikums im Rahmen der Fallstudie adäquat reflektieren zu können, werden die Ontologie-Konzepte aus OntHoS geeignet angepasst (Bastian 2005,

98; Crosby 2005, 85-86). Diese Erweiterungen erfolgten auf der Basis der bei *Bastian* (2005) und *Crosby* (2005) durchgeführten Analysearbeiten und sind somit bedarfsgerecht fundiert. Für das Verständnis der folgenden Ausführungen ist die detaillierte und umfassende Darstellung der Ontologie nicht erforderlich. Deshalb wird auf ihre ausführliche Beschreibung verzichtet.

Konzept	Beschreibung
Temporal Concept	Beschreibung von Zeitpunkten, -intervallen, oder einer Zeitdauer
Medical Concept	Beschreibung von medizinischem Wissen
Employee Concept	Beschreibung des Personals z.B. hinsichtlich Qualifikation oder Rollen
Appraisal Concept	• Beschreibung von Bewertungen anhand von absoluten oder relativen Kriterien • Grundlage für das Scheduling
Appointment Concept	Beschreibung von Terminvereinbarungen wie z.B. Zeitpunkt, erforderliche Personen oder Ressourcen
Document Concept	Beschreibung des Formats und Inhalts von klinischen Dokumenten sowie Standards für Krankenhausinformationssysteme
Process Concept	Beschreibung von Prozessen als eine Sequenz von atomaren Aktionen und Alternativen im Prozessablauf
Object Concept	Beschreibung von realen Objekten und Personen wie Medikamente oder Patienten
Constraint Concept	• Beschreibung von Präferenzen und Restriktionen • Grundlage für das Scheduling
Insurance Concept	Beschreibung der Krankenversicherung

Tabelle 6.2-1: ***Beschreibung der OntHoS-Ontologiekonzepte***
Quelle: Eigene Darstellung, zusammengefasst aus der Beschreibung bei *Kirn et al.* (2006a, 205-207)

Die Ausführungen in den Abschnitten 6.2.1 sowie 6.2.3 mit 6.2.5 beschreiben die für den Anwendungsfall relevanten Ergebnisse der Anwendung des NDA für die Domänenanalyse. Die Resultate bilden wesentliche Voraussetzungen für die weiteren Darstellungen des Designs und der Implementierung des agentenbasierten Systems. Ergänzt werden die Rahmenbedingungen aus der Domäne durch technische Voraussetzungen, die im folgenden Abschnitt 6.3 beschrieben werden.

6.3 Beschreibung von technischen Rahmenbedingungen

6.3.1 Java-Plattformen

Applikationen, die mit der Programmiersprache Java[34] entwickelt werden, werden zur Ausführungszeit von einer virtuellen Maschine, der Java Virtual Machine, interpretiert. Dazu wird der Programmtext zunächst über einen Compiler in ein Plattform unabhängiges Format übersetzt. Dieser Bytecode kann prinzipiell auf jeder virtuellen Java-Maschine ausgeführt werden. Diese virtuelle Maschine stellt eine vollständige Laufzeitumgebung für Java-Anwendungen

[34] Ausführliche und umfassende Informationen zu Java können unter der folgenden Adresse abgerufen werden: http://java.sun.com, zugegriffen am 15.01.2007.

zur Verfügung. Diese Maschinen werden für gängige Hardware-Plattformen angeboten, insbesondere auch für Geräte mit eingeschränkten Ressourcen. Dabei kann die Java-Laufzeitumgebung in einer modularen Weise entsprechend der vorherrschenden Hardware-Performanz angepasst werden. Damit können Java-Programme auf einer Vielzahl unterschiedlicher Plattformen zur Verfügung gestellt werden. Diese Tatsache ist insbesondere bei heterogenen Informationssystemlandschaften von Vorteil, wie sie im Gesundheitswesen zu beobachten sind. Deshalb wird für die Auswahl einer Agenten-Plattform als Basis für die in dieser Arbeit beschriebene, prototypische Realisierung auch die Implementierung in der Sprache Java vorausgesetzt (siehe dazu auch Abschnitt 6.3.2).

In Abbildung 6.3-1 sind alle verfügbaren Java-Plattformen der Version 2 dargestellt. Diese werden im Folgenden kurz beschrieben:

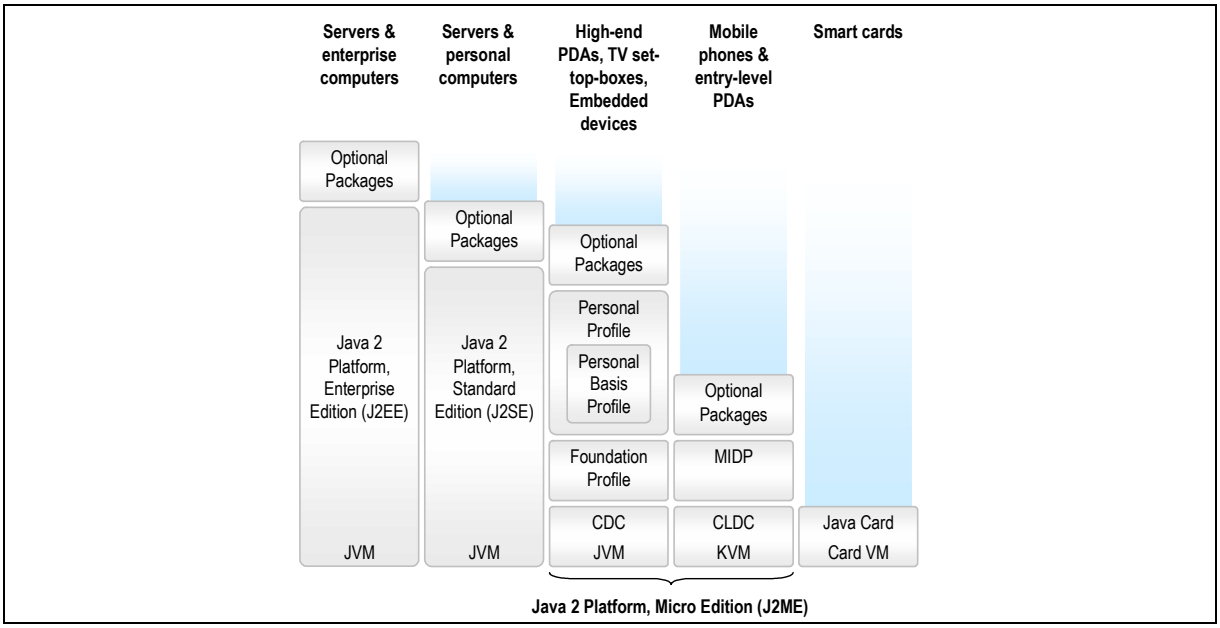

Abbildung 6.3-1: ***Überblick über Java-Plattformen***
Quelle: *Sun Microsystems Inc.* (2002, 1)

Die J2EE™-Version ist insbesondere für Anwendungen auf Servern (Hillebrand 2006, 34) ausgelegt. Durch optionale Pakete (Optional Packages) kann die Funktionalität der Bibliotheken bei allen Plattformen bis auf Java Card™ VM ergänzt werden. Diejenigen Teile, die Funktionalität auf der Server-Seite implementieren, fehlen bei der Version J2SE™. Letztere wird hauptsächlich für Anwendungen auf Arbeitsplatzrechnern eingesetzt.

Weil die Hardware-Performanz bei mobilen Endgeräten sowie eingebetteten Systemen im Vergleich zu stationären Systemen eingeschränkt ist, wird für die Version J2ME™ nur ein beschränkter Umfang der Leistungsfähigkeit der virtuellen Maschine zur Verfügung gestellt (Schmatz 2004, 7; Hillebrand 2006, 35). Weiterhin sind dabei die API- und die Sprachfunktionalität von Java reduziert (Schmatz 2004, 7). In Abhängigkeit des jeweiligen Endgerätes kann als Grundlage von J2ME eine JVM oder KVM (K Virtual Machine) eingesetzt werden. Weil die Hardware von mobilen Endgeräten sehr mannigfaltig gestaltet ist (siehe auch Hillebrand 2006, 35), ist bei der J2ME-Edition die Anpassbarkeit an die Anforderungen bzw.

die gegebene Hardware nötig. In Abbildung 6.3-1 sind die entsprechenden Konfigurationsmöglichkeiten dargestellt: Auf der Grundlage einer virtuellen Maschine werden Konfigurationen (CDC, Connected Device Configuration und CLDC, Connected Limited Device Configuration), Profile (Foundation Profile, Personal Profile, Personal Basis Profile und MIDP, Mobile Information Device Profile) sowie weitere optionale Pakete differenziert. Diese werden in Anlehnung an die Ausführungen bei *Sun Microsystems Inc.* (2002, 2) im Folgenden beschrieben:

Die CDC-Konfiguration umfasst im Gegensatz zur CLDC-Konfiguration eine vollständige virtuelle Maschine und einen höheren Anteil der J2SE-Funktionalität. Die Funktionalität der CLDC-Konfiguration ist entsprechend der zu erwartenden Hardware-Performanz eingeschränkt.

Über ein Profil werden zur Laufzeitumgebung weitere Funktionalitäten z.B. zur Realisierung der Benutzerschnittstelle oder zum Zugriff auf spezielle Eigenschaften der Geräte hinzugefügt. Zusammen mit CLDC wird mit dem Profil MIDP eine vollständige Java-Laufzeitumgebung zur Verfügung gestellt. Neben MIDP können für CLDC auch die Profile IMP (Information Module Profile) und IMP-NG (Information Module Profile – Next Generation) eingesetzt werden (Schmatz 2004, 9; Hillebrand 2006, 38). Das in Abbildung 6.3-1 exemplarisch genannte Foundation Profile ist das am wenigsten umfangreiche Profil und stellt deshalb nur rudimentäre Funktionalität ohne Benutzerinteraktion zur Verfügung. Das Personal Profile ergänzt das Foundation Profile z.B. um die vollständige Unterstützung der Java-Bibliothek Abstract Window Toolkit (AWT) zur Realisierung von grafischen Benutzeroberflächen. Personal Basis Profile ist eine Teilmenge von Personal Profile und unterstützt deshalb nur eine eingeschränkte Funktionalität für z.B. grafische Benutzeroberflächen. Die optionalen Pakete erweitern die Funktionalität speziell auf die Bedürfnisse der Anwendungen und können auch proprietäre Funktionalität wie z.B. Datenbank-Konnektivität zur Verfügung stellen.

Die Ausführungen zur Java-Laufzeitumgebung zeigen, dass mit der Wahl einer solchen Umgebung Plattformunabhängigkeit erreicht werden kann. Insbesondere erlaubt die Java-Plattform eine Konfiguration der Laufzeitumgebung und der zur Verfügung stehenden Funktionalitäten entsprechend den gegebenen Hardware-Ressourcen. Damit ist die Java-Plattform als Grundlage für ein Informationssystem im Gesundheitswesen prädestiniert, weil in dieser Domäne heterogene Systeme beobachtet werden können.

6.3.2 JADE als Implementierungsplattform

In diesem Abschnitt werden nach der Begründung für die Wahl der Plattform JADE das Architekturmodell dieser Plattform sowie relevante Kommunikationsmechanismen beschrieben:

Als Implementierungsbasis wird in der vorliegenden Arbeit die FIPA-konforme (siehe Abschnitt 2.3.4.1) Agentenplattform JADE[35] (Java Agent Development Framework, Telecom Italia Lab 2006) gewählt. Gründe hierfür können wie folgt zusammengefasst werden:

- Implementierung auf der Basis von Java
- Konformität zum Standard FIPA

[35] Siehe dazu http://jade.tilab.com, zugegriffen am 23.08.2006.

- Interoperabilität mit anderen Agentensystemen
- Breite Akzeptanz in der Wissenschaft (siehe dazu Paulussen et al. 2003a; Kirn et al. 2006b)
- Gleichzeitiger Einsatz als Agentenplattform und Implementierungsrahmenwerk einschließlich geeigneter Werkzeuge
- Bereitstellung umfangreicher Zusatzbibliotheken (siehe Abschnitte 6.3.3 mit 6.3.5 und 7.4)
- Integration der Ontologiemodellierung durch automatische Generierung von JADE-kompatiblem Java-Code über das Werkzeug Protégé (siehe dazu Abschnitt 2.3.7)
- Effizienz der Plattform (siehe dazu Abschnitt 6.3.6)

JADE ist auf Basis der Plattform unabhängigen Java-Laufzeitumgebung (siehe Abschnitt 6.3.1) implementiert und deshalb durch heterogene Hardware-Plattformen, wie sie im Gesundheitswesen oftmals vorgefunden werden, nicht eingeschränkt. Weiterhin unterstützt JADE den Entwicklungsprozess für ein agentenbasiertes System: JADE ist einerseits als Rahmenwerk implementiert, das als Grundlage für die Implementierung eingesetzt werden kann. Andererseits umfasst JADE eine Reihe von Werkzeugen (Bellifemine et al. 2005, 26-36), die den Entwicklungsprozess und das Testen der Anwendung unterstützen.

Aus den genannten Gründen wird in der vorliegenden Arbeit die Plattform JADE gewählt, deren Architektur und grundlegende Kommunikationsmechanismen im Folgenden beschrieben werden:

Das Architekturmodell von JADE ist in Abbildung 6.3-2 dargestellt und wird im Weiteren erläutert: JADE kann wegen der Java-Basierung auf verschiedenen Endgeräten mit unterschiedlicher Performanz ausgeführt werden. Vorausgesetzt wird eine der Laufzeitumgebungen J2EE, J2SE oder J2ME (siehe dazu Abschnitt 6.3.1). Eine Agentenplattform besteht aus einer Familie von Containern, welche auf mehrere Rechner verteilt sein können. Ein einziger Container einer solchen Plattform ist als Main Container ausgezeichnet. Dieser enthält neben dem Agent Management System (AMS) das Dienstverzeichnis Directory Facilitator (DF) und den Remote Monitoring Agent (RMA). Das AMS realisiert das Lebenszyklusmanagement der auf der Plattform existierenden Agenten und einen Namensdienst, um zu gewährleisten, dass jeder Agent einen eindeutigen Namen besitzt, und fungiert als Autorität innerhalb der Plattform (Caire 2003, 5). Im DF werden die registrierten Dienstbeschreibungen von Agenten abgelegt, um sie bei der Suche von Agenten nach geeigneten Diensten weiterzugeben (Caire 2003, 5). Der RMA repräsentiert die grafische Benutzeroberfläche zum Management der Agentenplattform und ermöglicht den Aufruf von Entwicklungswerkzeugen wie z.B. einen Sniffer, der den Inhalt der versendeten ACL-Nachrichten visualisiert und somit einen Beitrag zur Qualitätssicherung der Agentenanwendung leistet.

Eine Multiagentenanwendung kann, wie die Ausführungen weiter oben zeigen, über mehrere Endgeräte verteilt sein. Innerhalb einer Plattform ist die Migration von Agenten von einem Container zu einem anderen möglich. Zwischen verschiedenen Plattformen wird in der

aktuellen Version von JADE[36] keine Mobilität unterstützt. Über eine zusätzliche Erweiterung[37] kann dennoch eine Plattform übergreifende Mobilität von Agenten realisiert werden.

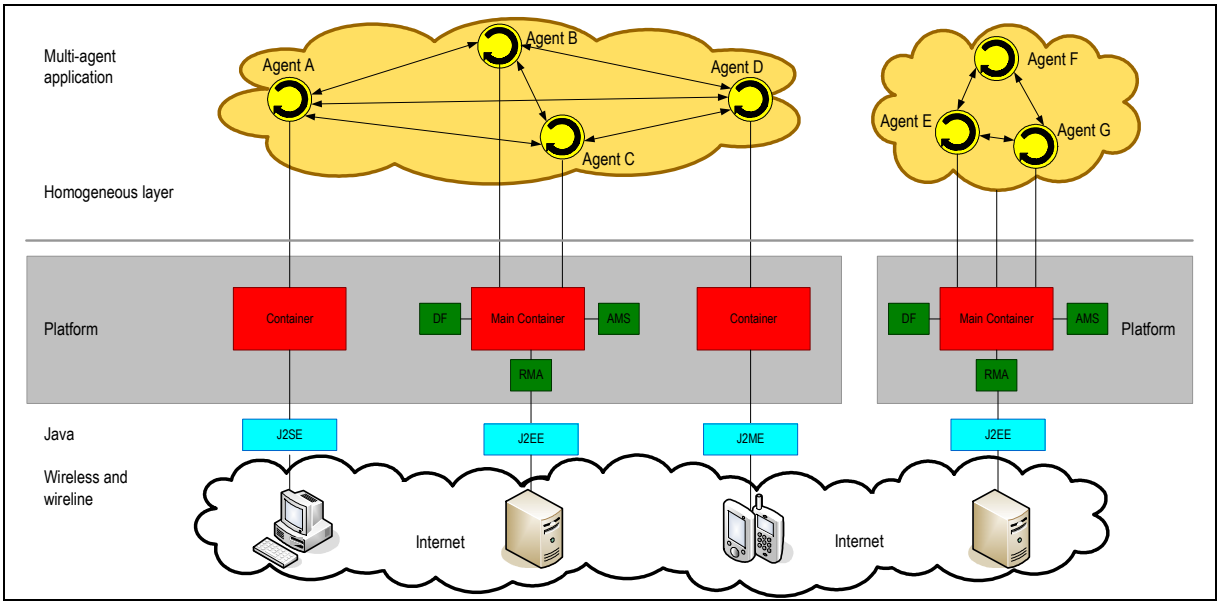

Abbildung 6.3-2: ***JADE-Architekturmodell***
Quelle: In Anlehnung an *Caire* (2003, 5) und *Caire* (2004, 4)

Die Kommunikation zwischen den auf die Endgeräte verteilten Containern erfolgt über drahtgebundene oder -lose Netzwerke. Über die Agentenplattform wird dabei eine Kommunikationsschicht zur Verfügung gestellt, um die Implementierungsarbeiten von Kommunikationsaufgaben zu abstrahieren.

Der abstrakte Kommunikationsmechanismus zwischen Agenten innerhalb einer JADE-Plattform ist in Abbildung 6.3-3 dargestellt und wird im Folgenden beschrieben: Die Grundlage für die Kommunikation bildet der asynchrone Nachrichtenaustausch (Caire 2003, 14). Nach der Vorbereitung einer Nachricht durch den Sender (Agent A1) wird diese über die JADE-Laufzeitumgebung an die Nachrichtenwarteschlage des Empfängers (Agent A2) weitergeleitet. Sobald dort Nachrichten eintreffen, wird der Empfängeragent benachrichtigt; der Zeitpunkt der Nachrichtenverarbeitung und ob die Nachricht verarbeitet wird, werden durch den Entwickler bestimmt (Caire 2003, 14).

Auf technischer Ebene wird bei der Kommunikation von Agenten bzw. Diensten zwischen drei Formen unterschieden, die aber konzeptionell nicht differenziert werden, sondern nur in ihrer Implementierung (die bei der folgenden Aufzählung jeweils angegebenen Kommunikationstechnologien werden bei Cortese/Quarta/Vitaglione 2002 genannt) unterschiedlich sind:

- Kommunikation zwischen Agenten auf unterschiedlichen Plattformen (IIOP, HTTP)
- Kommunikation zwischen Agenten in verschiedenen Containern auf der gleichen Plattform (RMI)

[36] Zum Zeitpunkt der Erstellung der vorliegenden Arbeit liegt JADE in Version 3.4 vor.
[37] Siehe dazu http://jade.tilab.com/doc/tutorials/migration.html, zugegriffen am 24.08.2006.

- Kommunikation zwischen Agenten innerhalb des gleichen Containers (Events)

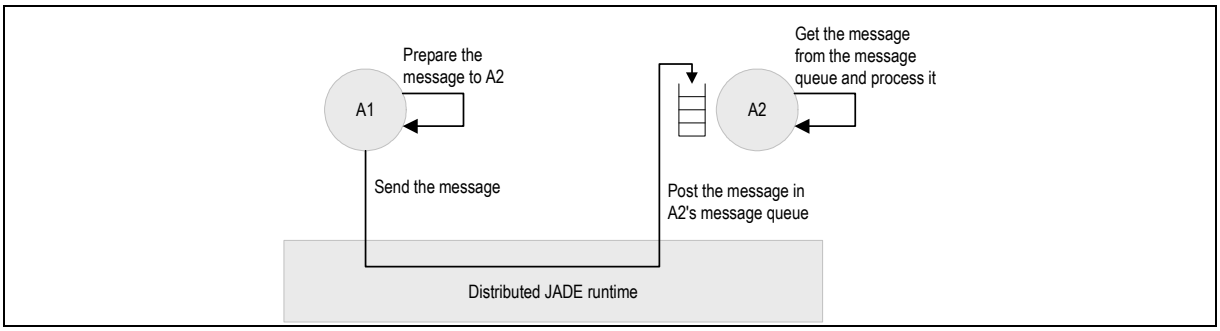

Abbildung 6.3-3: ***Asynchroner JADE-Kommunikationsmechanismus zwischen Agenten***
Quelle: *Caire* (2003, 14)

Obwohl damit unterschiedliche Kommunikationsszenarien identifiziert werden können, können alle über das im Folgenden beschriebene Konzept realisiert werden. Abbildung 6.3-4 stellt den Kommunikationsmechanismus dar, wie er bei jeder Kommunikation zwischen Agenten bzw. Diensten zu finden ist:

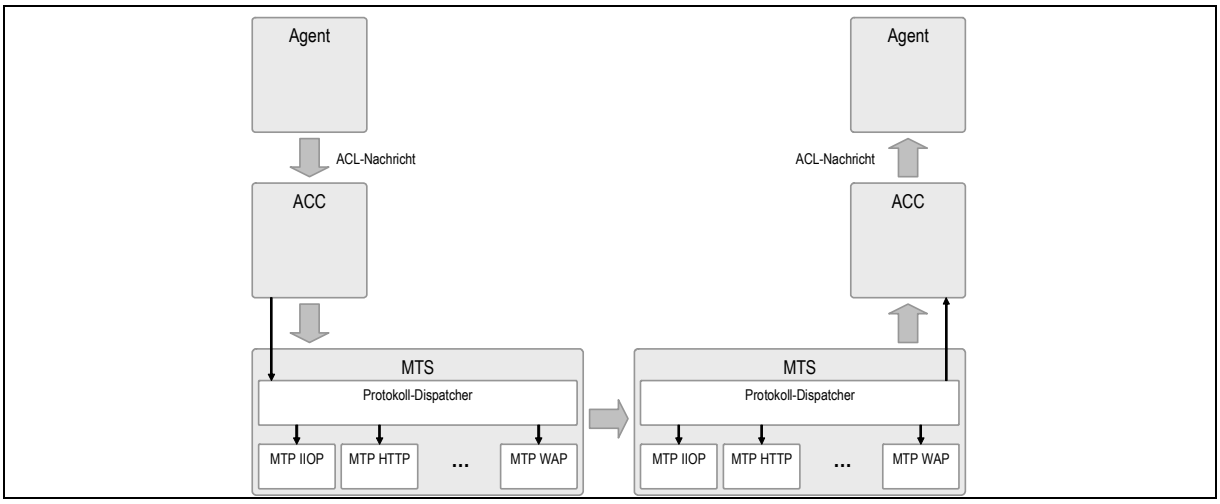

Abbildung 6.3-4: ***JADE-Kommunikation zwischen Agenten bzw. Diensten***
Quelle: Eigene Darstellung

Eine ACL-Nachricht eines Agenten wird zum Versand an den Agent Communication Channel (ACC) weitergeleitet. Der ACC stellt somit eine Schnittstelle zwischen einem Agenten und einem anderen Agenten bzw. einem Dienst dar. Vom ACC gelangt die Nachricht an die MTS-Komponente (Message Transport Service). Diese ist in die zwei Teile Protokoll-Dispatcher und eine Menge von MTP-Komponenten (Message Transport Protocol) gegliedert. Letztere kapseln die zur Verfügung stehenden Protokolle wie z.B. IIOP, HTTP oder WAP (Wireless Application Protocol) und versenden die Nachricht über das spezifizierte Protokoll. Der Nachrichteninhalt wird vom jeweiligen MTP zum gleichwertigen MTP des Empfängers versendet. Der Protokoll-Dispatcher des Absenders stellt dabei anhand der Adresse des Empfängers fest, welches Kommunikationsprotokoll der Empfänger verwendet, und verpackt den Nachrichteninhalt in einen geeigneten Umschlag. Für die Empfängeradressen kann alternativ eine Reihe unterschiedlicher Adressen angegeben werden. Kann der Empfänger

über die aktuell selektierte Adresse nicht erreicht werden, wird vom Protokoll-Dispatcher die nächste Adresse aus der Adressmenge ausgewählt. Diese Auswahl und ihre Reihenfolge kann jedoch nicht explizit durch den Anwendungsentwickler beeinflusst werden.

Auf der Empfängerseite wird die Nachricht vom Ziel-MTP angenommen und an den Protokoll-Dispatcher weitergeleitet. Letzterer entpackt die Nachricht aus dem ursprünglichen Umschlag und leitet die ACL-Nachricht über den ACC an den Empfänger weiter.

In diesem Abschnitt konnte nur ein beschränkter Überblick über die Eigenschaften der JADE-Plattform gegeben werden. Diese Beschreibung ist für das Verständnis der weiteren Ausführungen jedoch ausreichend. Eine ausführliche Darstellung zur Entwicklung von MASen auf der Basis von JADE ist bei Bellifemine/Caire/Greenwood (2007) beschrieben.

In den folgenden Abschnitten 6.3.3 mit 6.3.5 werden Erweiterungen der JADE-Plattform beschrieben, die Funktionalitäten für mobile Endgeräte, Sicherheit und die Integration von Web Services zur Verfügung stellen.

6.3.3 JADE-LEAP als Agentenplattform für mobile Endgeräte

6.3.3.1 Einschränkungen mobiler Endgeräte

Obwohl die Potenziale für mobile Endgeräte (siehe Abschnitt 4.6.8) auf Effektivitäts- und Effizienzfortschritte im Behandlungsprozess schließen lassen, sind mit diesen Geräten besondere Rahmenbedingungen zu berücksichtigen, die bei der Implementierung bzw. Portierung einer Anwendung auf ein mobiles Endgerät beachtet werden müssen. Diese Einschränkungen ergeben sich insbesondere aus der geringen Bauform sowie der Mobilität der Endgeräte (Müller-Wilken 2001, 15-18; Hillebrand 2006, 22-23):

- *Ressourcenknappheit*: Mobile Geräte werden meist über Akkus oder Batterien versorgt, weshalb die Nutzung dieser Ressourcen sparsam erfolgen muss. Auch der Prozessor ist hinsichtlich Größe und Stromverbrauch zu optimieren. Folglich kann dieser nur beschränkte Verarbeitungsfähigkeiten besitzen. Auch die Speicherkapazität von mobilen Geräten ist im Vergleich zu stationären Rechnern eingeschränkt.
- *Eingeschränkte Sicherheit*: Mobile Geräte unterliegen besonderen Sicherheitsrisiken, da drahtlos übertragene Daten von Dritten abgehört werden können, lokal gespeicherte Daten bei fehlender Stromversorgung nicht mehr verfügbar sein können, das Gerät an sich verloren, entwendet oder bei mobilen Einsätzen zerstört werden kann (Hillebrand 2006, 22-23).
- *Variierende Kommunikationsbedingungen*: Mobile Geräte benutzen für Kommunikationsvorgänge mit ihrer Umwelt meist drahtlose Netzwerkverbindungen wie IrDA® (Infrared Data Association), Bluetooth, WLAN (Wireless LAN), GPRS (General Packet Radio Service) oder UMTS (Universal Mobile Telecommunications System, Hillebrand 2006, 23). Durch schwankende Signalstärken, Interferenzen oder externe Störeinflüsse kann es dabei zu verringerten Übertragungsraten oder Verbindungsabbrüchen kommen (Hillebrand 2006, 23). Ein weiteres Problem stellt die dynamische IP-Adressenvergabe dar (Caire/Lhuillier/Rimassa 2002).
- *Eingeschränkte Benutzerschnittstelle*: Durch die angestrebte geringe Größe der mobilen Geräte müssen die Benutzerschnittstellen (Anzeigeeinheit, Eingabemöglichkeiten via Tastatur oder Stift) möglichst Platz sparend gestaltet werden. Bei der Entwicklung von

Anwendungen ist deshalb auf eine adäquate Realisierung von Benutzerschnittstellen zu achten.

- *Heterogenität*: Die Entwicklung von Geräten mit der Anforderung der Optimierung der Auslastung der beschränkten Ressourcen führt oft dazu, dass auf besondere Kompatibilitätsschichten verzichtet wird (Hillebrand 2006, 23). Deshalb sind mobile Anwendungen meist nicht auf andere mobile Endgeräte ohne Anpassungen übertragbar.

Wegen dieser Einschränkungen sind bei der Portierung der virtuellen Patientenakte auf ein mobiles Endgerät besondere Rahmenbedingungen zu berücksichtigen. Diese fließen auch in die in Abschnitt 6.7.3.2 über das Design der mobilen Anwendung beschriebenen Architekturentscheidungen ein.

6.3.3.2 Darstellung von JADE-LEAP

Da die Leistungsfähigkeit von mobilen Endgeräten für den Einsatz der JADE-Plattform oftmals nicht ausreicht (für eine detaillierte Begründung siehe Hillebrand 2006, 30), wird für solche Anwendungsszenarien die JADE-LEAP-Erweiterung für Geräte mit eingeschränkten Ressourcen eingesetzt. JADE-LEAP (Lightweight Extensible Agent Plattform) stellt die erste FIPA-konforme Plattform für mobile Endgeräte (Berger/Bauer/Watzke 2001; Berger et al. 2002; Berger/Müller/Seitz 2005, 15) einschließlich vollständiger JADE-Kompatibilität dar (Berger/Bauer/Watzke 2001) und kann auf Java fähigen Mobiltelefonen, Plattformen wie PDAs, stationären Rechnern und J2SE-Servern zum Einsatz kommen (Hillebrand 2006, 30). Um dieses Spektrum zu unterstützen, kann JADE-LEAP auf die folgenden Java-Laufzeitumgebungen angepasst werden (siehe auch Caire 2006; Hillebrand 2006, 30-31):

- *J2SE*: Ausführung von JADE-LEAP auf PCs und Servern mit J2SE[38] und drahtgebundener Kommunikation
- *PJava*: Unterstützung von JADE-LEAP auf Geräten mit PersonalJava bzw. der Nachfolgelaufzeitumgebung J2ME zusammen mit dem Personal Profile (siehe dazu auch Abschnitt 6.3.1)
- *MIDP*: Ausführung von JADE-LEAP auf Geräten mit dem MIDP-Profil (siehe dazu Abschnitt 6.3.1).

Diese drei Varianten sind in ihrer Implementierung zwar verschieden, bieten jedoch dieselben APIs an und verbergen somit die Implementierungsdetails vor dem Entwickler, die durch die Spezifika der jeweiligen Geräte und ihre Netzwerkverbindungen entstehen (Caire 2006, 8; Hillebrand 2006, 31). Auf der Basis dieser Laufzeitumgebungen werden von JADE-LEAP die folgenden beiden Ausführungsmodi unterstützt (siehe auch Caire 2006, 6; Hillebrand 2006, 31-32):

- *Stand-alone Ausführungsmodus*: Der Container einer Agentenplattform wird ausschließlich auf dem mobilen Endgerät ausgeführt. Der Main Container wird auf dem stationären System ausgeführt. Seit der JADE-Version 3.3 wird diese Ausführungsmöglichkeit nicht mehr weiterentwickelt und gewartet (Hillebrand 2006, 31). Deshalb wird bei *Caire* (2006) eine weitere Verwendung nicht empfohlen.

[38] Für diesen Anwendungsfall wird bei J2SE Version 1.4 oder höher vorausgesetzt (Hillebrand 2006, 30).

- *Split Ausführungsmodus (siehe Abbildung 6.3-5)*: Der Container wird in die zwei Komponenten FrontEnd (Ausführung auf dem mobilen Gerät) und BackEnd (Umgebung auf dem J2SE-Rechner) aufgeteilt. Vor dem Entwickler werden diese Details verborgen (Caire 2006, 8), somit muss auf der Implementierungsebene nicht zwischen den beiden Ausführungsmodi differenziert werden. Der Split Ausführungsmodus eignet sich besonders für mobile Geräte mit Ressourcenbeschränkung, da das FrontEnd leichtgewichtiger ist als der vollständige Container, die Startphase der gesamten Anwendung verkürzt wird und weniger Daten über die drahtlose Netzwerkverbindung zu versenden sind (Hillebrand 2006, 32). Voraussetzung für einen fehlerfreien Betrieb ist, dass beide Teilplattformen durch eine permanent bestehende Netzwerkverbindung verbunden sind, wobei auch die besonderen Eigenschaften von mobilen Geräten hinsichtlich Netzwerkkonnektivität berücksichtigt werden (siehe dazu die Ausführungen weiter unten in diesem Abschnitt).

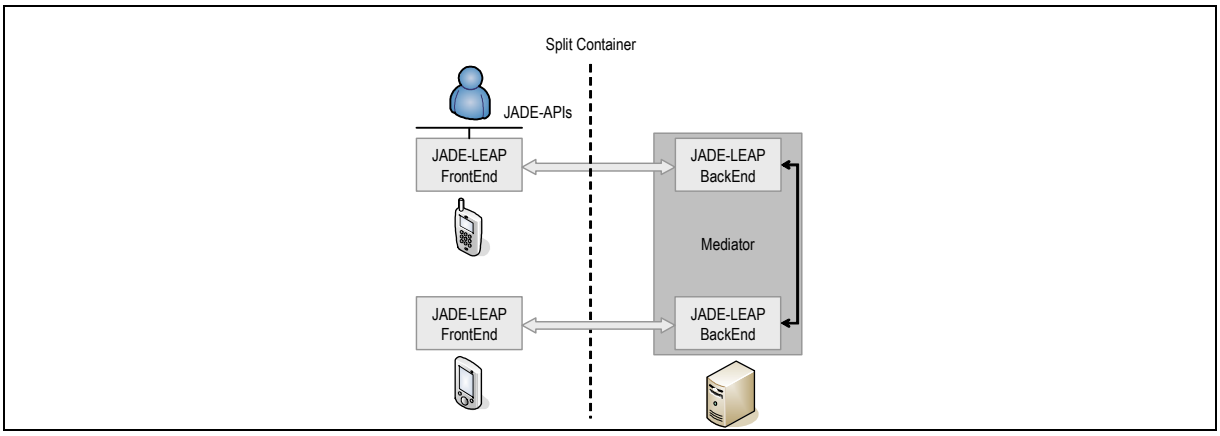

Abbildung 6.3-5: ***Split-Ausführungsmodus einer JADE-LEAP Plattform***
Quelle: An Anlehnung an *Caire* (2006, 8)

Zu der genannten Kommunikation zwischen dem FrontEnd und dem BackEnd wird das dediziert implementierte JICP-Protokoll (JADE Internal Communication Protocol, Caire/Lhuillier/Rimassa 2002) eingesetzt, welches auf TCP/IP basiert (Hillebrand 2006, 71). Je nach Gerät und dessen technischen Voraussetzungen gibt es dafür eine Reihe von Möglichkeiten (Bluetooth, WLAN, GRPS oder UMTS, Hillebrand 2006, 108), um die Verbindung für die Kommunikation zwischen FrontEnd und BackEnd aufzubauen. Mit der genannten, Mediator basierten JICP-Lösung in JADE-LEAP können die besonderen Eigenschaften von mobilen Endgeräten hinsichtlich Netzwerkkonnektivität berücksichtigt werden (Caire/Lhuillier/Rimassa 2002): Temporäre Verbindungsabbrüche können maskiert werden, indem die während des Verbindungsabbruches eingetroffenen Nachrichten in einem Puffer vorgehalten und bei der Verbindungswiederaufnahme abgerufen werden können. Auch ein dynamischer Wechsel der IP-Adresse beim Wiedereintritt des mobilen Gerätes in das Netzwerk wird mit diesem Ansatz vor den übrigen Anwendungen verborgen.

Neben den bereits beschriebenen Eigenschaften von JADE-LEAP können, angelehnt an die Ausführungen bei *Berger/Müller/Seitz* (2005, 15-16), weitere Charakteristika dieser besonderen Plattform identifiziert werden:

Um weitere Anforderungen von mobilen Endgeräten zu berücksichtigen, wurden im Vergleich zur herkömmlichen JADE-Version bei JADE-LEAP die Protokolle der Transportschicht angepasst: Anstatt der RMI-basierten Kommunikation innerhalb einer Agentenplattform, also zwischen verschiedenen Containern, wird das interne Protokoll TP (Transport Protokoll) verwendet. Die zugrunde liegenden Kommunikationskanäle wurden für den Einsatz von WLAN und GPRS ergänzt. Eine zusätzliche Erweiterung für JADE-LEAP unterstützt den Aufbau von Ad-hoc-Netzwerken über Bluetooth und WLAN: Existieren Plattformen innerhalb der durch die technischen Rahmenbedingungen vorgegebenen Kommunikationsreichweite, verbinden sich die Plattformen zu einer Gesamtplattform. Der Austausch von Nachrichten innerhalb dieses virtuellen Netzwerks und zwischen den realen Agentenplattformen erfolgt durch die Etablierung eines dynamischen Lookup-Dienstes (Pirker/Berger/Watzke 2004, 16). Dieser Dienst auf der Basis der JADE-Directory Facilitators ist in einer weiteren Ausbaustufe von JADE-LEAP durch eine dienstbasierte Suche in Ad-hoc-Netzwerken erweitert (Pirker/Berger/Watzke 2004). Dazu wurde die existierende Discovery Middleware JXTA[39] in die FIPA-Architektur integriert.

Mit JADE-LEAP wird eine auf die beschränkten Ressourcen von mobilen Endgeräten zugeschnittene Lösung der JADE-Plattform zur Verfügung gestellt. Dabei werden insbesondere auch wechselnde Netzwerkkonnektivitäten berücksichtigt.

6.3.4 Berücksichtigung von Sicherheitseigenschaften in der JADE-Plattform

Die wesentlichen Ziele der IT-Sicherheit sind die Vertraulichkeit, Verfügbarkeit und Integrität (Bundesminister der Justiz 1992, 1). Im Folgenden ist zu untersuchen, ob die gewählte Agentenplattform JADE diese Eigenschaften besitzt. Dazu werden einerseits die Funktionalitäten der Erweiterung JADE-S sowie relevante Eigenschaften von JADE beschrieben:

Insbesondere bei Agentensystemen als verteilten Systemen ist Sicherheit zu berücksichtigen (siehe dazu auch Abschnitt 2.3.10), wobei dazu erforderliche Funktionalitäten von dem Zusatzmodul JADE-S realisiert werden. Im Folgenden wird ein Überblick über dieses beschrieben:

In der aktuellen Version von JADE-S[40] werden folgende Funktionalitäten unterstützt (JADE Board 2005, 5-7):

- *Authentifizierung*: Nur berechtigte Benutzer können die JADE-Plattform starten. Diese Funktionalität basiert auf der JAAS[41] (Java Authentication and Authorization Service) API.
- *Berechtigungen*: Berechtigte Nutzer der JADE-Plattform erhalten bestimmte Rechte, um Aktionen für Agenten zu initiieren, Plattformdienste auszuführen oder Zugriff auf Anwendungsressourcen zu erhalten. Diese Berechtigungen bestimmen z.B. den Zugriff auf lokale Ressourcen wie die JVM™, das Dateisystem oder das Netzwerk. Die gewünschten Berechtigungen werden in einer Datei nach der JAAS-Syntax abgelegt. Da-

[39] Weitere Informationen können auf den folgenden Seiten abgerufen werden: http://www.jxta.org/, zugegriffen am 26.02.2007.

[40] Die zum Zeitpunkt der Erstellung der vorliegenden Arbeit aktuelle Version ist JADE-S 2.

[41] Siehe dazu http://java.sun.com/products/jaas, zugegriffen am 24.08.2006.

mit werden Berechtigungen sowohl im Main Container als auch in den übrigen Containern festgelegt.

- *Nachrichtenintegrität und -vertraulichkeit*: JADE-S unterstützt die Verschlüsselung und Signatur für ACL-Nachrichten. Dieser Sicherheitsmechanismus für ACL-Nachrichten wird wie folgt umgesetzt: Eine ACL-Nachricht ist aus einem Umschlag mit Kommunikationsinformationen wie Absender und Empfänger und dem eigentlichen Nachrichteninhalt aufgebaut. Der Inhalt wird verschlüsselt und signiert. Die Signatur wird im Umschlag hinzugefügt. Die Verschlüsselung und Signierung des Nachrichteninhalts erfolgt für den Benutzer transparent. Letzterer muss nur angeben, dass die Nachricht verschlüsselt bzw. signiert werden soll. Wenn Fehler beim Verschlüsseln, Signieren, Entschlüsseln oder Verifizieren auftreten, wird die gesamte Nachricht verworfen und an den Sender über das AMS als Fehlernachricht (FAILURE) zurückgesendet. Mit diesem Mechanismus wird gewährleistet, dass die bei einem Agenten ankommenden Nachrichten korrekt verschlüsselt, signiert, entschlüsselt und verifiziert wurden. Für diesen Zweck besitzt jeder Agent ein Paar aus einem privaten und öffentlichen Schlüssel (JADE Board 2005, 4).

Die Realisierung von Verschlüsselung und Signatur auf Implementierungsebene wird im Folgenden beschrieben (angelehnt an JADE Board 2005, 13-14): Über die Methode getHelper der Klasse Agent kann die Klasse SecurityHelper referenziert werden. Ihre Methoden setUseEncryption und setUseSignature veranlassen die Plattform, die Nachrichten zu verschlüsseln und zu signieren. Für die Verschlüsselung der Kommunikation zwischen zwei Agenten ist ein Principal-Objekt desjenigen Agenten erforderlich, der die verschlüsselte Nachricht empfangen soll. Ein solches Objekt kann für die Verschlüsselung von Nachrichten zwischen unterschiedlichen Plattformen in der aktuellen Version der zur Verfügung stehenden JADE-API[42] nur einer solchen Nachricht entnommen werden, die vom Empfänger der zu verschlüsselnden Nachricht signiert wurde. Für diesen Zweck werden für die Verschlüsselung von Nachrichten zwischen unterschiedlichen Plattformen in der vorliegenden Arbeit die folgenden Schritte durchgeführt:

- Versand einer Nachricht an den Empfänger durch den Versender der zu verschlüsselnden Nachricht
- Empfangen der Nachricht und Versenden der signierten Nachricht
- Empfangen der signierten Antwort des Empfängers beim Sender
- Extraktion des öffentlichen Schlüssels aus der Nachricht
- Versand der verschlüsselten Nachricht an den Empfänger durch den Versender

JADE-S ist insbesondere mit JADE-LEAP (siehe Abschnitt 6.3.3) kompatibel (JADE Board 2005, 3) und daher auch auf mobile Anwendungen übertragbar. Trotz der bisher dargestellten, in JADE-S implementierten Funktionalitäten werden die folgenden Aspekte von der aktuellen Version[43] nicht unterstützt (JADE Board 2005, 4):

- Fehlende Unterstützung für mobile Agenten

[42] Siehe dazu die Anmerkung in Fußnote 36.
[43] Siehe dazu die Anmerkung in Fußnote 40.

- Der Informationsaustausch zwischen zwei Containern erfolgt unsigniert über SSL (Secure Sockets Layer). Ein Angreifer könnte damit einen horizontalen Nachrichtenaustausch von einem Container zum anderen unterwandern.
- Die meisten Sicherheitseigenschaften stehen für Agenten auf MIDP-Geräten nicht zur Verfügung.

Weil im Main Container in der JADE-Plattform wesentliche Dienste wie z.B. der DF angesiedelt sind, würde der Ausfall des Main Containers das Agentensystem stark einschränken (Bellifemine et al. 2006, 20). Deshalb wird von JADE die Replikation des Main Containers (siehe Abbildung 6.3-6) unterstützt, um die Verfügbarkeit zu erhöhen (siehe für die folgende Darstellung die Ausführungen bei Bellifemine et al. 2006, 19-24):

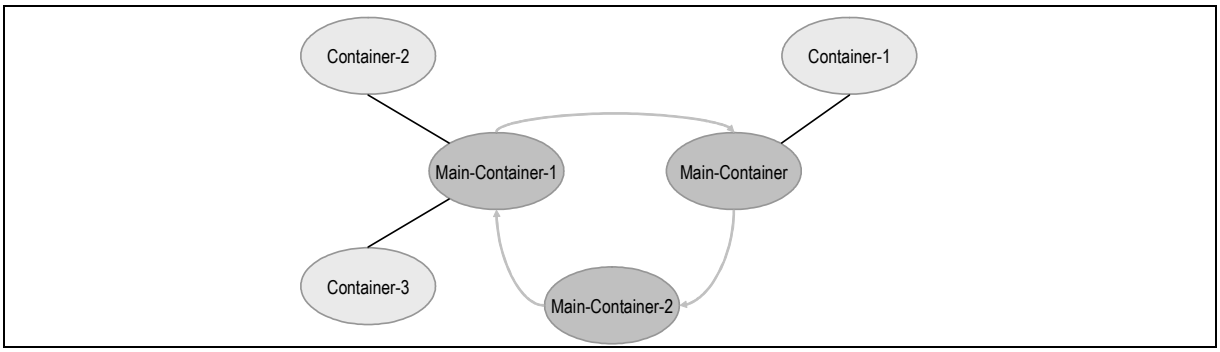

Abbildung 6.3-6: ***Replikation von Main Containern der JADE-Plattform***
Quelle: In Anlehnung an *Bellifemine et al.* (2006, 20)

Dazu bilden die replizierten Main Container eine Ringstruktur und synchronisieren sich wechselseitig. Die Container der Plattform können mit beliebigen verfügbaren Main Containern assoziiert sein, wobei gewährleistet sein muss, dass ein Container genau einem Main Container zugeordnet ist. Die tatsächlich vorhandene Menge der Main Container ist für den jeweiligen Container bei Abwesenheit von Fehlern transparent. Fällt ein Main Container aus, wird dies vom im Ring vorhergehenden Main Container registriert und an die übrigen Main Container kommuniziert. Als Folge besteht der Ring der Main Container aus einer geringeren Anzahl von Main Containern. Wenn der mit einem Container assoziierte Main Container ausfällt, wird dies vom jeweiligen Container registriert und ein alternativer Main Container gesucht, der einer Liste der in seiner Umgebung verfügbaren Main Container entnommen wird.

Die am Anfang dieses Abschnitts genannten wesentlichen Ziele der IT-Sicherheit können mit der Funktionalität von JADE-S bzw. JADE umgesetzt werden. Vertraulichkeit und Integrität der Nachrichten wird über die Verschlüsselung bzw. die Signatur[44] *ermöglicht. Die Verfügbarkeit wird durch die Einführung von Redundanzmechanismen erreicht, welche gewährleisten, dass die Infrastrukturdienste in einer Agentenplattform verfügbar bleiben.*

[44] Trotz der Darstellung durch das *JADE Board* (2005, 6), nach der die Verschlüsselung und Signatur von Nachrichten auch zwischen verschiedenen Agentenplattformen unterstützt würden, wurde in der in Abschnitt 6.8.3 beschriebenen Implementierung festgestellt, dass lediglich die Signatur zwischen unterschiedlichen Plattformen möglich ist. Der Austausch von verschlüsselten Daten zwischen Agenten auf differenzierten Agentenplattformen konnte hingegen nicht erfolgreich umgesetzt werden.

Die Problemstellungen, die sich aus der potenziellen Mobilität von Agenten ergeben, werden von JADE-S explizit nicht berücksichtigt. Bei Kamel Boulos et al. (2006) wird ein Lösungsansatz (siehe dazu auch Abschnitt 2.3.10 für eine ausführliche Darstellung) beschrieben, der zwar auf Mobilität von Agenten verzichtet, aber dennoch geeignete Mechanismen zur Verfügung stellt, die die Sicherheit in einem Agentensystem erhöhen und gleichzeitig die gewünschte Funktionalität von mobilen Agenten gewährleisten können.

6.3.5 JADE Web Service Integration Gateway

Weil Web Services (siehe Abschnitt 3.2.4.4), insbesondere im Hinblick auf Service orientierte Architekturen (siehe Abschnitt 3.2.4.2) eine bedeutende Stellung einnehmen, und das Paradigma der Agentenorientierung auch Elemente der Dienstorientierung beinhaltet, ist es naheliegend, die beiden Ansätze zu kombinieren, um Web Services basierte Anwendungen in ein Agentensystem integrieren zu können. Die dafür durch eine JADE-Erweiterung verfügbare Funktionalität wird im Folgenden beschrieben:

Nach dem Wrapper-Prinzip (Jennings 2001, 40) können auch Web Services über eine Erweiterung der JADE-Plattform in ein Agentensystem integriert werden. Dazu wurde das JADE Web Service Integration Gateway (WSIG, Greenwood 2005) entwickelt. Dabei wird eine bidirektionale Kommunikation zwischen Agentendiensten und Web Services auf der Basis von WSDL/SOAP/UDDI realisiert (Greenwood 2005, 4). In den Standard-Dienstverzeichnissen DF (für ein Agentensystem) und UDDI (für Web Services) werden die Dienste zum Registrieren, Deregistrieren, Modifizieren und Suchen angeboten (Greenwood 2005, 4).

Die für die Funktionalitäten zur Integration von Web Services in Agentensysteme erforderliche Architektur (siehe auch Abbildung 6.3-7) wird im Folgenden beschrieben. Dazu werden insbesondere die Prozesse zum Registrieren und Aufrufen von Diensten dargestellt. Diese Ausführungen sind angelehnt an die Darstellung bei *Greenwood* (2005, 1-18):

Das Gateway (siehe Abbildung 6.3-8) nimmt Registrierungs- und Deregistrierungsanforderungen für Dienstbeschreibungen sowie Suchanfragen für Dienste von Agenten und Web Services entgegen. Diese Anfragen werden geeignet verarbeitet und die Dienstbeschreibungen in den internen Registern adäquat abgelegt: ACL-Dienstbeschreibungen werden im DF der Agentenplattform abgelegt, die WSDL-Beschreibungen für Web Services im UDDI-Dienst. Die Dienstbeschreibungen in der Sprache ACL bzw. WSDL werden dazu mit der Komponente ServiceDescriptionTranslator (siehe Abbildung 6.3-8) in die jeweils komplementäre Dienstbeschreibung transformiert und im entsprechenden Dienstverzeichnis abgelegt: Die WSDL eines zu registrierenden Web Services wird vom GatewayAgent aus dem tModel[45] extrahiert, welches im UDDI registriert ist. Außerdem wird vom GatewayAgent die WSDL nach FIPA ACL/SL0[46] übersetzt und eine entsprechende Dienstbeschreibung generiert sowie im DF re-

[45] Ein tModel besitzt die folgende Aufgabe (OASIS UDDI Specifications TC 2002, 7): In einem tModel werden Referenzen zu Informationen angegeben, die die technischen Eigenschaften eines Web Services beschreiben. Damit ist ein tModel ein Metamodell, welches definiert, wie die technischen Spezifikationen eines Web Services gestaltet sind. Die Spezifikationen zu UDDI und insbesondere auch zum tModel können auf den folgenden Seiten abgerufen werden: http://www.uddi.org, zugegriffen am 26.02.2007.

[46] SL0 wird nach *The Foundation for Intelligent Physical Agents (FIPA)* wie folgt definiert (siehe dazu auch die Ausführungen bei The Foundation for Intelligent Physical Agents 2002, 14):
„Profile 0 is denoted by the normative constant fipa-sl0 in the language parameter of an ACL message. Profile 0 of FIPA SL is the minimal subset of the FIPA SL content language.

gistriert. Die eigentliche Übersetzung zwischen ACL-Dienstbeschreibungen und WSDL-Beschreibungen sowie umgekehrt wird von der Komponente ServiceDescriptionTranslator vorgenommen.

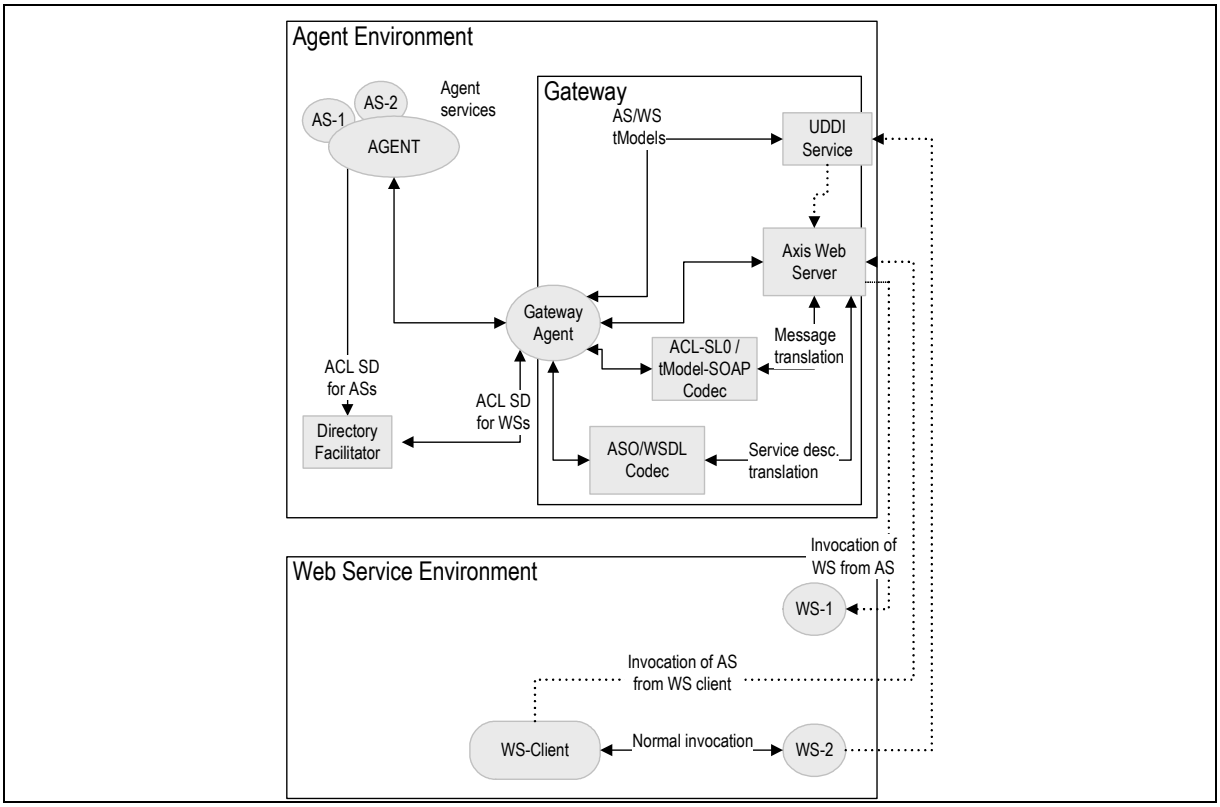

Abbildung 6.3-7: ***WSIG-Architekturmodell***
Quelle: In Anlehnung an *Greenwood* (2005, 6)

Die Interaktionen bei der Registrierung, dem Aufruf eines von einem Agenten angebotenen Dienst über Web Services sowie und umgekehrt werden im Folgenden beschrieben:

Wenn Agenten ihre Dienste im DF registrieren, wird vom DF zunächst überprüft, ob eine Agentendienstbeschreibung mit einer zusätzlichen Web Services-Dienstbeschreibung vorliegt. Ist diese Bedingung erfüllt, wird der Diensteintrag zum GatewayAgent propagiert. Dort wird vom GatewayAgent eine Übersetzung des Dienstes von ACL/SL0 nach WSDL in der Komponente ServiceDescriptionTranslator (siehe Abbildung 6.3-8) angestoßen, ein entsprechendes tModel für UDDI generiert und dieses im UDDI registriert. Abschließend wird aus der WSDL-Beschreibung ein Stub-Objekt für den Agentendienst in der SOAP-Implementierung Apache Axis[47] generiert. Über diese Stub-Objekte können die Agentendienste von Web Services aufgerufen werden.

It allows the representation of actions, the determination of the result a term representing a computation, the completion of an action and simple binary propositions".

[47] Informationen zum Axis Web Server können auf den folgenden Seiten abgerufen werden: http://ws.apache.org/axis/, zugegriffen am 25.05.2007.

Der Aufruf eines Agentendienstes über die Web Services-Schnittstelle wird im Folgenden dargestellt: Der Web Services Client identifiziert im WSIG UDDI einen passenden Dienst. Nach der Identifikation des Dienstes wird ein entsprechender SOAP-Aufruf abgesetzt. Die darauf folgenden Aktionen werden von dem GatewayAgent ausgeführt: Nach dem Empfang der Nachricht wird die WSDL-Beschreibung extrahiert. Wenn sich dieser Dienst im UDDI befindet, wird eine entsprechende ACL-Nachricht konstruiert. Wenn keine Antwort auf den ursprünglichen Aufruf des Web Services erwartet wird, wird die Nachricht an den entsprechenden Agenten zur Weiterverarbeitung gesendet. Andernfalls wird zusätzlich die entsprechende Antwort des aufgerufenen Agentendienstes abgewartet und daraus eine adäquate SOAP-Antwort für den ursprünglichen Sender generiert. Diese Kommunikation wird über ein Stub-Objekt mediiert.

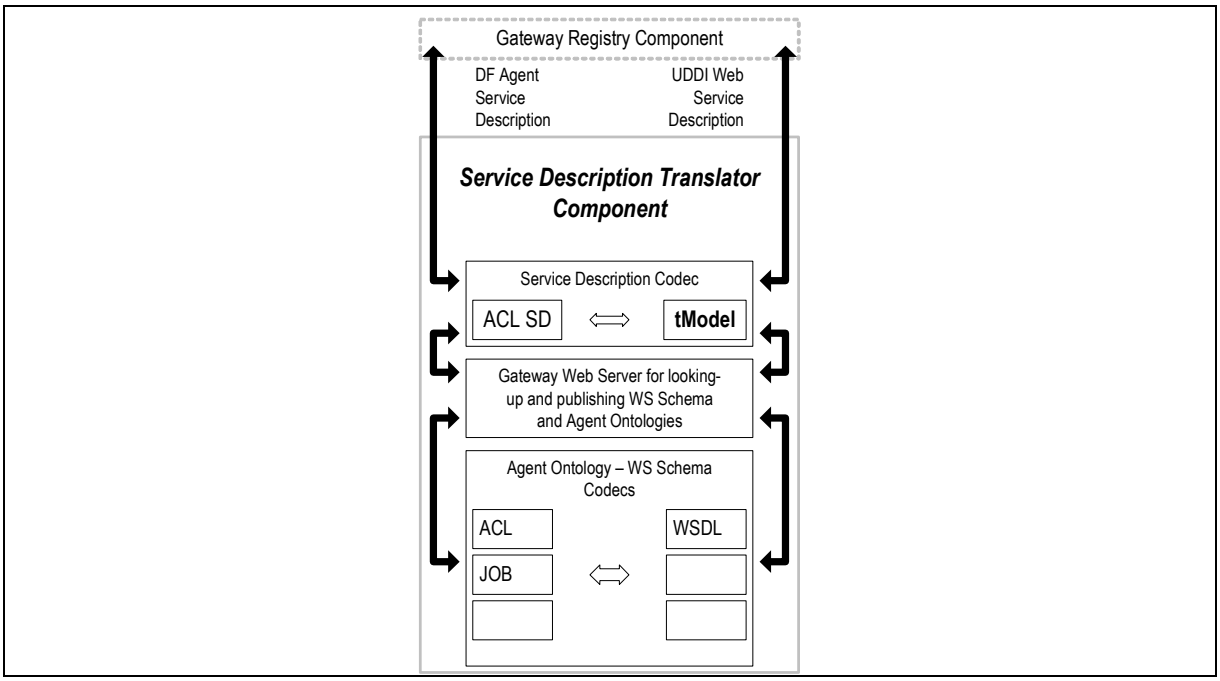

Abbildung 6.3-8: ***Übersetzung zwischen den heterogenen Dienstbeschreibungen ACL SD und WSDL***
Quelle: *Greenwood* (2005, 9)

Der Aufruf eines Web Service-Dienstes über einen Agenten erfolgt nach dem folgenden Interaktionsmuster: Der benötigte Dienst wird von einem Agenten im DF der Plattform identifiziert. Eine entsprechende FIPA-Nachricht wird konstruiert und an den GatewayAgent versandt. Die nun folgenden Schritte werden vom GatewayAgent ausgeführt: Die empfangene Nachricht des aufrufenden Agenten enthält eine Anforderung für einen Web Service-Aufruf. Aus dieser Nachricht wird die ACL-Dienstbeschreibung extrahiert. Wenn diese Dienstbeschreibung im DF vorgehalten wird, wird die entsprechende WSDL Dienstbeschreibung im UDDI tModel zurückgeliefert. Die SOAP-Nachricht wird auf der Basis dieser WSDL-Dienstbeschreibung erstellt. Wenn keine Antwort auf diese Anfrage erwartet wird, wird die Nachricht an den Empfänger versendet. Andernfalls wird zusätzlich ein lokales Stub-Objekt generiert und dann die SOAP-Nachricht versandt. Sobald die Antwort des aufgerufenen Web Services im Stub-Objekt vorliegt, wird aus der empfangenen SOAP-Nachricht eine ACL-Nachricht konstruiert.

Diese wird an den ursprünglichen Aufrufer des Dienstes gesendet. Schließlich wird das temporäre Stub-Objekt entfernt.

Die WSIG-Erweiterung erlaubt die Integration von Web Services in eine Agentenplattform. Außerdem kann diese Integration grundsätzlich auch auf mobile Endgeräte übertragen werden: Dazu ist eine entsprechende WSIG-Version mit Unterstützung für mobile Endgeräte in Planung, wobei diese Variante insbesondere Verbindungsabbrüche unterstützt (Greenwood 2005, 18) und somit die besonderen Eigenschaften von mobilen Endgeräten berücksichtigt.

Für das in der vorliegenden Arbeit angestrebte Integrationsszenario im Gesundheitswesen ist eine ausreichende Performanz der JADE-Agentenplattform erforderlich. Im folgenden Abschnitt 6.3.6 wird deshalb die Leistungsfähigkeit von JADE beleuchtet.

6.3.6 Performanz der JADE-Plattform

Die Effizienz einer Agentenplattform stellt für das in dieser Arbeit vorgestellte Konzept eine wesentliche Anforderung dar. Denn die Abbildung jedes Benutzers oder Patienten durch einen Agenten sowie die funktionale Dekomposition von Agenten führen zu einer umfangreichen Anzahl von Agenten in einer Plattform (Chmiel et al. 2005, 160). Dies wiederum bedingt eine entsprechend hohe Anzahl von zwischen Agenten ausgetauschten Nachrichten (Chmiel et al. 2005, 160). In einer Untersuchung betrachten *Chmiel et al.* die Kriterien Anzahl von Agenten bzw. Nachrichten sowie die Effizienz bei der Agentenerzeugung und -migration (2005, 160). In mehreren Experimenten wurden dazu die folgenden wechselnden Rahmenbedingungen untersucht:

- Anzahl der Agenten
- Anzahl und Größe der ausgetauschten Nachrichten
- Anzahl, Ausstattung und Heterogenität der eingesetzten Rechner
- Anzahl der Container auf einer Plattform
- Anzahl der Migrationen
- Anzahl der dynamisch erzeugten Agenten

Insgesamt konnte in den Experimenten festgestellt werden, dass die JADE-Plattform robust und effizient ist sowie linear skaliert (Chmiel et al. 2005, 171). Insbesondere konnten *Chmiel et al.* folgende Aussagen treffen:

- JADE skaliert sehr gut bei der Betrachtung der Anzahl der ausgetauschten Nachrichten (Chmiel et al. 2005, 162).
- Die Performanz skaliert linear hinsichtlich der Migration von Agenten (Chmiel et al. 2005, 163-165).
- Die Leistungsfähigkeit von JADE skaliert bei einem Szenario mit der Erzeugung von Agenten linear (Chmiel et al. 2005, 170).

Darüber hinaus konnten als weitere Ergebnisse von *Chmiel* (2005, 166 und 171) die folgenden Resultate gefunden werden:

Insgesamt kann festgestellt werden, dass die Performanz der JADE-Plattform hinsichtlich Verarbeitungsgeschwindigkeit bei gleichzeitiger Erhöhung der Anzahl der Agenten bzw. der

zwischen ihnen ausgetauschten Nachrichten sowie der durchgeführten Migrationen linear skaliert.

Weiterhin kann auf der Grundlage der durchgeführten Experimente konstatiert werden, dass die Limitierungen der Performanz durch die Einschränkungen der Programmiersprache Java entstehen, die in einer virtuellen Maschine interpretiert und ausgeführt wird. Dabei sind die Prozessorverarbeitungsgeschwindigkeit, die Hauptspeichergröße sowie die Geschwindigkeit des zugrunde liegenden Netzwerkes für die erreichbare Performanz entscheidend.

Als Nebenergebnis konnte festgestellt werden, dass eine JADE-Plattform auf einer Maschine mit dem Betriebssystem Solaris™ hinsichtlich der auf einer Plattform möglichen Anzahl von Agenten performanter ist als eine JADE-Plattform auf einer Maschine mit dem Betriebssystem Windows. Somit ist aus Performanzgründen die Bevorzugung einer Betriebssystemplattform gegenüber einer anderen möglicherweise sinnvoll.

Die Ergebnisse der Untersuchungen zeigen zusammenfassend, dass die JADE-Plattform für die Entwicklung von umfangreichen agentenbasierten Systemen geeignet ist. Dies lässt sich damit begründen, dass die in einem Agentensystem inhärent umfangreiche Anzahl von Agenten, ihren Migrationen, Generierungen und Nachrichten hinsichtlich der Verarbeitungsgeschwindigkeit mit linearem Aufwand skalierbar ist. Deshalb ist die gewählte JADE-Plattform als Grundlage für die in der vorliegenden Arbeit beschriebene Implementierung grundsätzlich ausreichend. Für einen Einsatz in der Praxis ist in weiteren Arbeiten zu eruieren, wie viele Agenten, repräsentiert durch Akteure im Gesundheitswesen und weitere Funktionalitäten implementierend, auf einer Agentenplattform allokiert werden können und wie viele Agentenplattformen erforderlich sind.

Ausgehend von den Darstellungen in den Abschnitten 5.4 und 6.3 werden im folgenden Abschnitt 6.3.6 die Anforderungen an die prototypische Implementierung konkretisiert.

6.4 Konkretisierung der umzusetzenden Anforderungen

Prinzipiell werden in der Software-Entwicklung funktionale und nicht funktionale Anforderungen unterschieden. Erstere beschreiben die Funktionalitäten, die das zu entwickelnde System zu leisten hat. Nicht funktionale Eigenschaften beziehen sich auf Nebenbedingungen, die z.B. die Performanz oder die Laufzeitumgebung betreffen. Für den zu entwickelnden agentenbasierten Prototyp werden im Folgenden die funktionalen (Abschnitt 6.4.1) und nicht funktionalen Anforderungen (Abschnitt 6.4.2) beschrieben. Es sollen dazu die in Abschnitt 5.4 beschriebenen Realisierungen des Anforderungskatalogs aus Abschnitt 4.12 weiter konkretisiert werden.

6.4.1 Funktionale Anforderungen

Als funktionale Anforderungen werden die folgenden Eigenschaften (für eine Zusammenfassung siehe Tabelle 6.4-1) festgehalten, die aus den bisherigen Ausführungen abgeleitet werden und in der prototypischen Implementierung der vorliegenden Arbeit geeignet umzusetzen sind:

Zur Einbettung in die TI für die eGK ist insbesondere zu zeigen, wie die Funktionalität dieser Architektur erweitert werden kann, um eine virtuelle Patientenakte auf der Basis der eGK zu etablieren und damit einen Mehrwertdienst anzubieten. Deshalb werden in der Implementierung die eGK und die für das Auslesen der Karten sowie für die Kommunikation zwischen

den verteilten Informationssystemen erforderlichen Konnektoren abgebildet. Weiterhin wird die Anforderung der Protokollierung der 50 letzten Zugriffe auf die eGK umgesetzt, um einen Ausschnitt aus den in der TI zu implementierenden Sicherheitseigenschaften zu demonstrieren.

Funktionale Anforderung	**Detaillierte Beschreibung**
Einbettung in die TI für die eGK (Anforderung 1 aus Abschnitt 4.12.1)	1. Implementierung der Konnektoren zum Lesen der Gesundheitskarten und zur Gewährleistung der Kommunikation zwischen den verteilten Informationssystemen 2. Bereitstellung von (simulierten) Kartenlesegeräten 3. Protokollierung der letzten 50 Zugriffe auf die eGK
Unterstützung des syntaktischen Standards HL7 CDA (Anforderung 2 aus Abschnitt 4.12.3)	4. Dokumentenaustausch auf der Basis von HL7 CDA Release 2
Einführung des AMD als Konstrukt der Software-Technik zur Unterstützung einer eindeutigen Patientenidentifikation (Anforderung 3 aus Abschnitt 4.12.4)	5. Kapselung von relevanten Daten im Konstrukt des AMDs entsprechend der EAI-Datenintegration
Einbettung des semantischen Standards LOINC in die ausgetauschten Dokumente (Anforderung 4 aus Abschnitt 4.12.6)	6. Einbettung von LOINC-Codes in das HL7 CDA-Dokument 7. Abbildung der Nachrichteninhalte auf geeignete Ontologiekonzepte
Informationsintegration aus verteilten und heterogenen Informationssystemen sowie Prozessunterstützung für einen ausgewählten Behandlungsprozess (Anforderung 5 aus Abschnitt 4.12.7)	8. Integration von SAP IS-H bzw. i.s.h.med über JCo und BAPIs 9. Agentenbasierte Prozessunterstützung
Demonstration der Flexibilität von Agentensystemen über ein dynamisches Scheduling-System sowie die Berücksichtigung der Patientenorientierung durch die Bereitstellung eines Patientenportals (Anforderung 6 aus Abschnitt 4.12.8)	10. Realisierung einer agentenbasierten Terminplanung 11. Realisierung der eKiosk-Funktionalität über die Bereitstellung eines Web basierten Patientenportals und seine Integration über die WSIG-Komponente in das MAS

Tabelle 6.4-1: ***Zusammenfassung der funktionalen Anforderungen***
Quelle: Eigene Darstellung

Aus der Perspektive von Primärsystemen ist die TI für den Aufruf von Diensten mit einer Pull-Funktionalität konstruiert (siehe dazu Neuhaus/Deiters/Wiedeler 2006, 334). Damit wird, unterstützt durch die Einrichtung einer geeigneten Firewall, verhindert, dass auf Informationssysteme der Leistungserbringer direkt zugegriffen werden kann, und gewährleistet, dass das

Sicherheitsmodell einfach gehalten wird (Neuhaus/Deiters/Wiedeler 2006, 334). Trotzdem ist mit dieser Architekturentscheidung eine Reihe von Nachteilen verbunden, die im Folgenden beschrieben werden:

Die Architekturgestaltung mit der Pull-Funktionalität erfordert die Verwaltung eines zentralen Servers, auf dem im Anwendungsfall der elektronischen Patientenakte die Daten der bisherigen Behandlungsgeschichte gespeichert werden müssen. Damit wächst einerseits der dort erforderliche Speicherplatz, weil für alle Patienten ausreichender Speicherplatz zur Verfügung gestellt werden muss. Andererseits werden die Daten für eine Institutionen übergreifende Patientenakte redundant vorgehalten, weil in den jeweiligen Primärsystemen die Daten der Patienten ohnehin gespeichert werden und verfügbar sein müssen. Weiterhin kann durch eine zentrale Speicherung der Daten die Akzeptanz durch die Patienten erschwert werden, weil von ihnen die Gefahr der unerlaubten Einsichtnahme der zentralen Daten angenommen werden könnte. Bei einem Zugriff durch einen Leistungserbringer auf ein Informationssystem außerhalb der eigenen Institution, wie er für die virtuelle Patientenakte erforderlich ist, wird durch die Vermeidung der Push-Funktionalität die Implementierung eines zusätzlichen Mehrwertdienstes erforderlich, der zur Realisierung der Push-Funktionalität zur Benachrichtigung beim Vorliegen von neuen Behandlungsergebnissen umzusetzen ist (siehe dazu auch Neuhaus/Deiters/Wiedeler 2006, 338-339).

Um die genannten Defizite der Pull-Funktionalität zu beseitigen, wird die in der vorliegenden Arbeit vorgestellte Patientenakte nicht auf dem beschriebenen Pull-Konzept aufgebaut, sondern virtuell und nur zur Laufzeit zusammengestellt. Dazu ist allerdings eine Push-Funktionalität aus Sicht der Primärsysteme erforderlich, weil auf diese zur Aggregation der virtuellen Patientenakte direkt zugegriffen werden muss und die Interaktion direkt zwischen den Primärsystemen ohne eine mediierende Komponente erfolgt. Obwohl das Sicherheitsmodell mit einer Pull-Funktionalität einfacher gehalten werden kann (Neuhaus/Deiters/Wiedeler 2006, 334), sind mit der Push-Funktionalität dennoch effektive Sicherheitsmechanismen umsetzbar (Sunyaev 2005, 29). Insofern stellt die Realisierung über die Push-Funktionalität keine Einschränkungen dar.

XML bietet die Möglichkeit, von heterogenen Plattformen zu abstrahieren und es stehen für XML etablierte Bibliotheken für die Verarbeitung und Transformation von XML-Dateien zur Verfügung (Krafzig/Banke/Slama 2006, 21-22). Die sich daraus begründende Flexibilität etabliert XML als den am meisten geeigneten Standard, der die Probleme der Applikations- und Middleware-Heterogenität lösen kann (Krafzig/Banke/Slama 2006, 22). Mit diesen Eigenschaften können XML basierte Dokumente automatisiert weiterverarbeitet und zusätzlich auch für Anwender visualisiert werden. Der offene, auf XML basierende Standard HL7 CDA kann aus diesen Gründen auch im Gesundheitswesen eine Etablierung erfahren. Insbesondere ist HL7 im Gesundheitswesen ein Standard, dem ein hohes Potenzial für die Interoperabilität beigemessen werden kann und der sich im stationären sowie in angepasster Form auch im ambulanten Sektor weit verbreitete. Aus den genannten Gründen wird in der vorliegenden Implementierung der Standard HL7 CDA Release 2 eingesetzt.

Eine eindeutige Patientenidentifikation muss dazu in der Lage sein, die unterschiedlichen Dokumente, die für einen Patienten im Laufe seiner Krankheitsgeschichte generiert werden, dem

zugehörigen Patienten zuzuordnen. Dazu ist ein entsprechendes Konstrukt auf der Ebene der Software-Technik erforderlich, welches von den möglicherweise noch in den unterschiedlichen Institutionen existierenden Patientenidentifikationen abstrahiert, indem Dokumente virtuell aus den verteilten Informationssystemen aggregiert werden. Für diese Anforderung ist das Konzept der aktiven medizinischen Dokumente geeignet, in denen Daten entsprechend der EAI-Datenintegration gekapselt werden. Das genannte Konzept wird ausführlich in Abschnitt 6.5.1 beschrieben.

Die Patientendaten, die sich im KIS SAP IS-H bzw. i.s.h.med des ausgewählten Referenzklinikums befinden, besitzen einen je nach Zweck definierten Code (Bastian 2005, 99). Dieser Code ist aus Gründen der Interoperabilität und des Verzichts auf proprietäre Nomenklaturen auf einen etablierten medizinischen Standard zu übertragen. Eine Möglichkeit besteht in der Transformation dieses Codes auf einen geeigneten LOINC-Code (Bastian 2005). Die Wahl des LOINC-Codiersystems in Kombination mit HL7 wird wie folgt begründet: UMLS (siehe dazu Abschnitt 4.11.3.3) zielt auf die Konsolidierung unterschiedlicher Standards zur Etablierung eines einheitlichen Wortschatzes ab. Dazu werden verschiedene Codiersysteme, Vokabulare oder Klassifikationen wie LOINC, HL7, SNOMED CT oder ICD-10 integriert. Um Inhalte in einem Agentensystem zwischen Agenten austauschen zu können, sind diese geeignet in die versendeten Nachrichten zu integrieren. Dazu wird eine entsprechend erarbeitete Ontologie eingesetzt (siehe dazu Abschnitt 6.2.5).

Das etablierte KIS SAP IS-H bzw. dessen klinische Erweiterung i.s.h.med wird als ein beispielhaftes Informationssystem ausgewählt, um Daten daraus zu extrahieren, die in die virtuelle Patientenakte integriert werden können. Der Zugriff auf die Daten aus dem SAP-System erfolgt über die Java-Bibliothek JCo[48] (Java Connector), über die die BAPI-Schnittstelle für die Geschäftsobjekte verfügbar gemacht werden kann. Die Wahl für die Anbindung über BAPIs liegt in der flexiblen Erweiterungsmöglichkeit durch eigene BAPIs auf der Grundlage der ABAP™-Programmierung entsprechend den jeweiligen Anforderungen.

Da zum Zeitpunkt der Erstellung der vorliegenden Arbeit kein geeignetes PVS zur Verfügung stand, wird der Zugriff auf dessen Funktionalität nur simuliert. Dies mindert die Qualität der Arbeit nicht, da mit der Anbindung des KISs bereits exemplarisch demonstriert wird, wie eine Integration gestaltet werden kann.

Aufgrund des Umfangs, aber auch zur Erreichung von Effizienzsteigerungen können definierte Pfade für standardisierte Behandlungsprozesse im Alltag der Leistungserbringer einen Beitrag leisten. Deshalb wird in der vorliegenden Arbeit auch beleuchtet, welche Eigenschaften eine geeignete Pfadunterstützung in einem agentenbasierten System besitzen muss.

Agentensysteme zeichnen sich insbesondere durch ihre Flexibilität aus. Deshalb ist in der vorliegenden Arbeit auch ein geeigneter Ausschnitt zu wählen, der dies demonstriert. Die Flexibilität ist diejenige Eigenschaft, die das Gesundheitswesen besonders auszeichnet. Auf Ebene

[48] Die Bibliothek JCo erlaubt die Integration von BAPI-Abfragen (siehe dazu Abschnitt 6.6.3.4) in ein Java-Programm. Weitere Informationen zu JCo finden sich auf den folgenden Seiten: http://help.sap.com, zugegriffen am 02.01.2007.

der Software-Technik kann diese Charakteristik geeignet durch Software-Agenten abgebildet werden (Kirn 2006). Besonders deutlich wird diese Anforderung bei Scheduling-Szenarien. Dazu ist in der vorliegenden Arbeit zu zeigen, dass nicht antizipierbare Ereignisse geeignet durch Software-Agenten verarbeitet werden können.

Neben der zunehmenden Integration im Gesundheitswesen kann die Tendenz der Patientenorientierung beobachtet werden (Waegemann 1999, 116). Deshalb ist auch in der in dieser Arbeit beschriebenen Implementierung ein Konzept zu erarbeiten, welches die Integration des Patienten berücksichtigt. Dazu wird ein Web basierter Ansatz auf der Basis von Web Services gewählt, weil dabei offene und etablierte Standards eingesetzt werden können. Weiterhin kann die Dienstorientierung sowohl in einer Agentenarchitektur als auch bei Web Services beobachtet werden. Aus diesen Gründen werden in dem in Abschnitt 6.3.5 beschriebenen Ansatz die beiden Konzepte Agentendienste und Web Services gekoppelt.

6.4.2 Nicht funktionale Anforderungen

Die funktionalen Anforderungen werden durch die folgenden nicht funktionalen Anforderungen ergänzt (siehe dazu auch Hillebrand 2006, 67-68) und sind in Tabelle 6.4-2 zusammengefasst:

Nicht funktionale Anforderung	**Detaillierte Beschreibung**
Unterstützung unterschiedlicher Plattformen und Endgeräte (Anforderung 1 aus Abschnitt 4.12.1)	12. Implementierung auf der Basis der Java-Laufzeitumgebung (Wahl begründet in Abschnitt 6.3.1) 13. Portierung des Prototyps zur Informationsintegration auf das ausgewählte mobile Endgerät vom Typ XDA II 14. Berücksichtigung der beschränkten Ressourcen des mobilen Endgerätes, insbesondere bei der Anpassung der Benutzeroberfläche
Architekturprinzipien (Anforderungen aus Abschnitt 5.3.2 zur Unterstützung der Anforderung 5 aus Abschnitt 4.12.7)	15. Berücksichtigung von etablierten EAI- und SOA-Prinzipien 16. Schichtenarchitektur 17. Modulare Implementierung zur Erweiterung mit zusätzlicher Funktionalität: Integration eines Simulationssystems und eines Patientenportals 18. Flexible Komposition und Dekomposition von Agentensubsystemen zur Informationsaggregation
Implementierung auf der Basis der etablierten Agentenplattform und des Rahmenwerks JADE (Anforderung aus Abschnitt 6.3.2)	19. Implementierung auf der Basis von JADE und seinen Erweiterungen

Berücksichtigung des Datenschutzes für die sensiblen medizinischen Daten (Anforderung 1 aus Abschnitt 4.12.1)	20. Einsatz der Erweiterung JADE-S

Tabelle 6.4-2: ***Zusammenfassung der nicht funktionalen Anforderungen***
Quelle: Eigene Darstellung

Wie in Abschnitt 6.3.1 ausgeführt, kann die Java-Laufzeitumgebung auf unterschiedlichen Hardware-Plattformen eingesetzt werden. Außerdem ist die Umgebung entsprechend der existierenden Performanz konfigurierbar. Somit eignet sich die Java-Laufzeitumgebung für das heterogene und mit unterschiedlichen Endgeräten sowie Hardware-Plattformen ausgestattete Gesundheitswesen.

Weil im Gesundheitswesen sowohl Patienten, Leistungserbringer, Daten und weiteres Arbeitsmaterial mobil sind (für Ausführungen zur Mobilität von Akteuren im Gesundheitswesen siehe Niemann/Eymann 2006; zur Mobilität von Akteuren und Arbeitsmaterial siehe Janz/Pitts/Otondo 2005), können mobile Endgeräte einen wesentlichen Beitrag dazu liefern, das informationslogistische Prinzip zu verstärken: Da diese Geräte, zumindest für einen beschränkten Zeitraum ohne Infrastruktur betrieben werden können, können diese an diejenigen Lokationen mitgeführt werden, an denen die Information benötigt wird. Dies ist oftmals mit stationären Geräten nicht realisierbar, allein schon aufgrund ihrer Größe.

Die Portierung nur der Anwendung zur Informationsintegration auf ein ausgewähltes mobiles Endgerät stellt keine Minderung der Ergebnisse der vorliegenden Arbeit dar. Denn mit der Portierung soll lediglich demonstriert werden, dass eine agentenbasierte Anwendung auch auf mobile Endgeräte mit eingeschränkten Ressourcen übertragen werden kann. Weitere Funktionalität kann entsprechend ergänzt werden.

Für die exemplarische Portierung wurde aus den folgenden Gründen ein PDA vom Typ XDA II[49] gewählt: Ein solches Gerät besitzt geringes Gewicht und einen geringen Größenumfang, eine Mitführung des PDA ist deshalb in der täglichen Arbeit der Leistungserberinger möglich. Zudem ist die Benutzerschnittstelle eines solchen Gerätes so gestaltet, dass auch noch umfangreichere Dokumente adäquat visualisiert werden können. Dazu sind im Vergleich zur stationären Anwendung die besonderen Voraussetzungen für die Implementierung der Benutzerschnittstelle zu berücksichtigen, um die Daten auf der beschränkten Anzeigefläche anzuzeigen. Da die compilierte Anwendung in dem Plattform unabhängigen Java Bytecode vorliegt, ist für die Bestimmung einer geeigneten Betriebssystemplattform für die mobile und die stationäre Anwendung keine besondere Restriktion gegeben. Lediglich die Verfügbarkeit von virtuellen Maschinen kann die Auswahl einschränken.

Das agentenbasierte Informationssystem ist so zu gestalten, dass es einerseits erweiterbar, andererseits auch mit anderen Applikationen interagieren kann. Dazu eignet sich eine Vorgehensweise auf der Basis von Architekturbausteinen. Die für die Konstruktion eingesetzte

[49] Informationen zu diesem Gerät können unter der folgenden Adresse abgerufen werden: http://www.my-xda.com/alreadyHaveII.html, zugegriffen am 30.09.2007.

ArBaCon-Methode (siehe dazu Abschnitt 1.7) wird gewählt, weil diese speziell mit der Anforderung der Architekturorientierung entwickelt wurde. Ein dabei berücksichtigtes Architekturprinzip ist die Schichtenarchitektur. Exemplarisch wird ein Simulationsprogramm an das Agentensystem angebunden, um die Erweiterungsfähigkeit zu demonstrieren. Außerdem wird die Applikation des Portalsystems geeignet an das modulare AMD-Konzept angekoppelt. Weiterhin sind in einem Integrationsszenario insbesondere EAI- und SOA-Konzepte geeignet zu berücksichtigen. Die Flexibilität von Agentensystemen kann bei der Komposition und Dekomposition von Agentensubsystemen zur Informationsaggregation demonstriert werden.

In Abschnitt 6.3.2 wurde bereits ausgeführt, weshalb sich JADE als Plattform für die Entwicklung eines Agentensystems eignet. Insbesondere die Entwicklung auf der Basis einer Open Source-Lizenz ermöglicht die Erweiterung durch zusätzliche Module, wie sie in den Abschnitten 6.3.3 mit 6.3.5 exemplarisch beschrieben wurden. Die Wahl für eine Agentenplattform fällt deshalb auf das FIPA-kompatible JADE.

Um die sensiblen Daten von Patienten in einem verteilten Agentensystem adäquat zu transportieren, sind geeignete Mechanismen in der Anwendung umzusetzen. Dazu eignet sich die Erweiterung JADE-S, weil damit, entsprechend den Ausführungen in Abschnitt 6.3.4, Sicherheitseigenschaften realisiert werden können.

Mit der Konkretisierung der Anforderungen zur Realisierung des Anforderungskatalogs aus den Abschnitten 4.12 und 5.4 ist in diesem Abschnitt die Voraussetzung für das Design (Abschnitte 6.5 mit 6.7) und die Konstruktion eines geeigneten agentenbasierten Systems (Abschnitt 6.8) gegeben. Damit wird die Beantwortung von Forschungsfrage 2 aus Abschnitt 1.4 vorbereitet.

6.5 Überblick über die Gesamtarchitektur

Ausgehend von der Beschreibung des für die Implementierung zentralen Elements der aktiven medizinischen Dokumente (Abschnitt 6.5.1) wird in Abschnitt 6.5.2 ein Gesamtüberblick über die Architektur der implementierten Anwendung gegeben.

6.5.1 Aktive medizinische Dokumente

Aktive medizinische Dokumente (AMD) sind das zentrale Element der im Folgenden beschriebenen Architektur zur Umsetzung einer virtuellen Patientenakte einschließlich erweiterter Funktionalität und werden im Weiteren erläutert:

Ein aktives medizinisches Dokument ist eine Aggregation von medizinischen und koordinativen Informationen (Anforderung 5 aus Abschnitt 6.4.1). Bisher eingesetzte Informationssysteme im Gesundheitswesen sollen mit diesem Konstrukt nicht ersetzt, sondern vielmehr integriert werden. Demnach werden Informationssysteme bspw. in einem Krankenhaus weiterhin zur Dokumentation z.B. von Befunden und Untersuchungsanforderungen eingesetzt. Ein AMD kann kontext- und benutzerabhängig Informationen zur Verfügung stellen, die ähnlich wie unterschiedliche Sichten auf ein Modell in der Modellorientierung betrachtet werden können. Ein AMD ist weiterhin mit genau einem Patienten assoziiert und existiert für einen Patienten in genau einer Ausprägung innerhalb einer Institution. Mit dem AMD werden alle bisher erfassten und vom Patienten zur Integration in seine Patientenakte freigegebenen Daten

zur Verfügung gestellt. Somit wird die Brücke zwischen unterschiedlichen Institutionen im Gesundheitswesen geschlagen. Folglich wird mit AMDen die Grundlage dafür geschaffen, dass Daten nach dem informationslogistischen Prinzip über Institutionsgrenzen hinweg den Leistungserbringern zur Verfügung gestellt werden können.

Mit dem AMD wird also eine zusätzliche Virtualisierungsschicht über die heterogenen Informationssysteme im Gesundheitswesen gelegt (siehe Abbildung 6.5-1). So wie durch institutionsinterne Integrationsarchitekturen (siehe dazu Abschnitt 4.6.1) heterogene Informationssysteme integriert werden können, wird durch die Virtualisierung durch ein AMD eine weitere Abstraktionsstufe in der Schichtenarchitektur hinzugefügt. Über diese Virtualisierungsschicht können Daten aus unterschiedlichen Institutionen, heterogenen Informationssystemen oder auch Archiven in dem AMD aggregiert werden. Damit stellt das AMD einen einzigen und zentralen Einstiegspunkt zur Navigation in den Patientendaten dar.

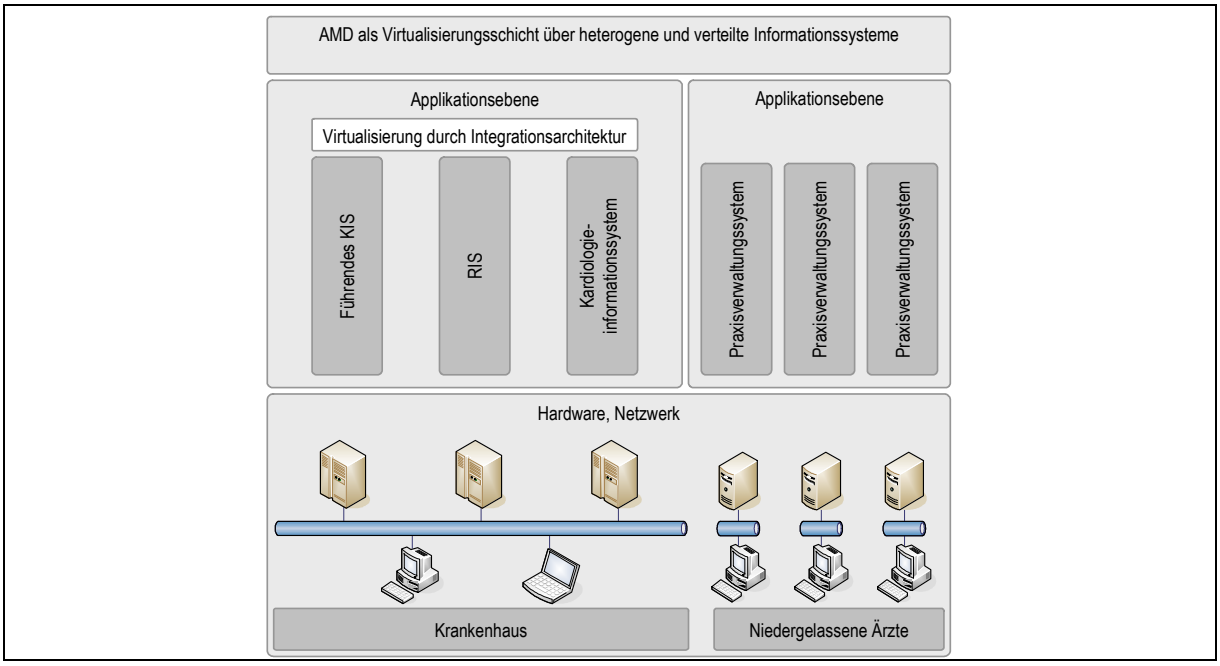

Abbildung 6.5-1: ***AMD als Virtualisierungsschicht***
Quelle: Eigene Darstellung

Das AMD ist realisiert als eine Kapselung von internen Agenten und ist selbst als Agent implementiert. Eine abstrakte Übersicht über die aktiven Elemente des AMDs ist in Abbildung 6.5-2 dargestellt. Diese Agenten bearbeiten jeweils spezielle Aufgaben und werden im Folgenden beschrieben:

Der AppointmentMonitor überwacht anstehende Termine und teilt diese den Anwendern mit. Im AppointmentManager werden Verhandlungen um Termine zwischen Patienten und Untersuchungseinheiten koordiniert. Der DataWrapper ist der singuläre Einstiegspunkt zu den Daten und Funktionalitäten des AMD. In diesem Agenten wird überprüft, welche Agenten mit dem DataWrapper interagieren dürfen. Im ViewManager werden die Daten für die Anzeige an der Benutzeroberfläche aufbereitet. Der DataRetrievalManager ist für die Steuerung der Datenabfrage aus verteilten Informationssystemen verantwortlich. Dazu werden für jedes Informations-

system spezielle Task-Agenten generiert, die die eigentlichen Anfragen an die Informationssysteme vornehmen. Nach der Abarbeitung und dem Empfang der Daten von den Task-Agenten werden diese von DataRetrievalManager zur Weiterverarbeitung über den DataWrapper an den ViewManager weitergeleitet. Der ProcessMasterAgent steuert die Abarbeitung von Behandlungsprozessen (Anforderung 9 aus Abschnitt 6.4.1) über Informationen, welche in einem zentralen Prozess-Repository abgelebt sind. Die Aufgabe des Subscribers ist die Ankopplung eines Web basierten Patientenportals. Das AMD ist somit als Agent mit mehren internen Agenten konstruiert und kann modular erweitert werden, indem neue dedizierte Agenten ergänzt werden.

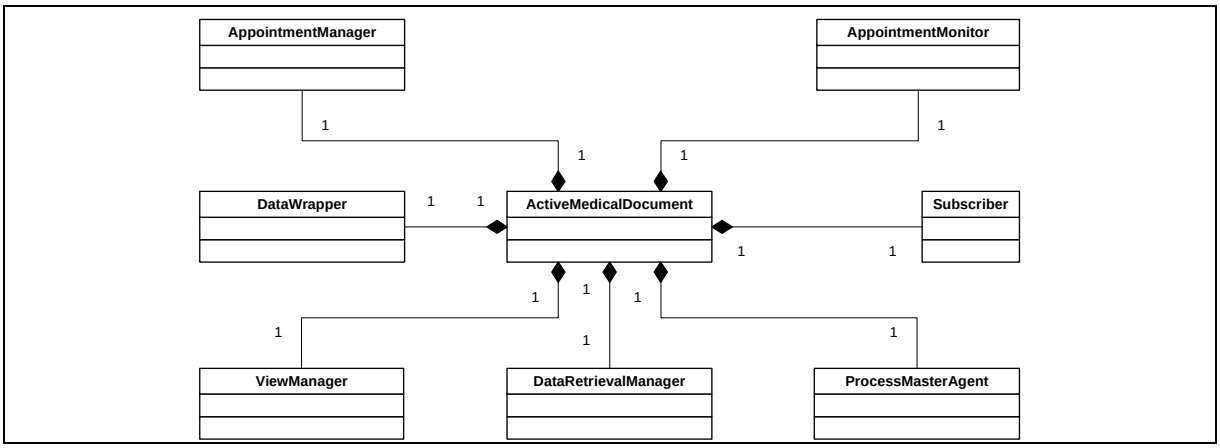

Abbildung 6.5-2: ***Überblick über die interne Struktur eines AMDs***
Quelle: Eigene Darstellung

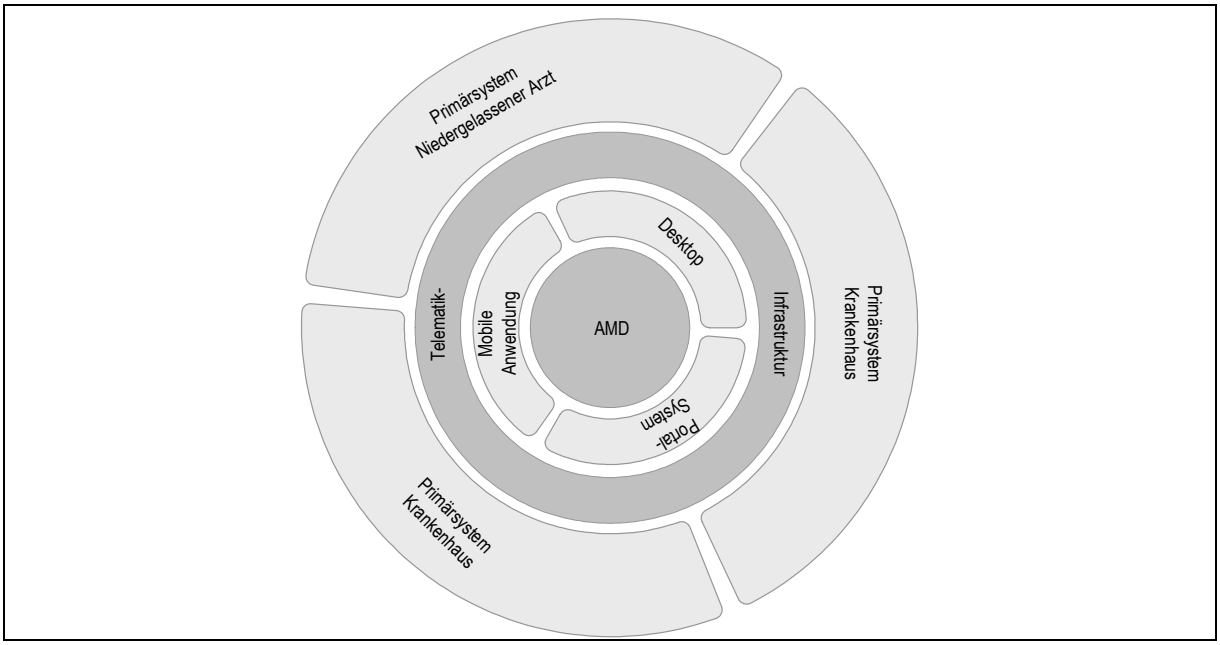

Abbildung 6.5-3: ***AMD im Zentrum unterschiedlicher Anwendungen***
Quelle: Eigene Darstellung

Das AMD bietet Schnittstellen zur Integration unterschiedlicher Anwendungen. Damit können verschiedene Applikationen mit dem AMD interagieren (siehe Abbildung 6.5-3). Im

Zentrum der in dieser Arbeit beschriebenen Implementierung steht dabei eine stationäre Lösung, deren Laufzeitumgebung sich üblicherweise auf einem Arbeitsplatzrechner befindet. In diese Anwendung ist auch die Prozesssteuerung integriert, womit Behandlungsprozesse auf der Basis von Software-Agenten automatisiert und kontrolliert durch den Anwender gesteuert werden können. Die stationäre Applikation wird durch eine mobile Anwendung ergänzt, die die Funktionalität der Informationsaggregation auf ein mobiles Endgerät portiert. Die Funktionalitäten des AMD können auch in ein Web basiertes Portalsystem integriert werden. In der vorliegenden Arbeit wurde dabei die Einsichtnahme in die virtuelle Patientenakte durch den Patienten gewählt. In das Portalsystem kann der Patient zusätzlich weitere Gesundheitsdaten eingeben. Damit wird die Patientenorientierung unterstützt, wie sie bei *Waegemann* (1999, 116) propagiert wird. Zur Virtualisierung der AMD-Schicht ist die Integration von Daten aus verteilten Informationssystemen erforderlich. Über die Kommunikationsmechanismen und Komponenten der TI werden dazu Primärsysteme angeschlossen, aus denen die Informationen für die virtuelle Patientenakte aggregiert werden.

Zusammenfassend kann festgestellt werden, dass in einem AMD neben Patientendaten auch koordinative (Anforderung 5 aus Abschnitt 6.4.1) sowie Informationen zur Steuerung von Behandlungsprozessen (Anforderung 9 aus Abschnitt 6.4.1) gekapselt werden. Damit wird die Daten- sowie Prozessintegration im Sinne der EAI (Anforderung 15 aus Abschnitt 6.4.2) erreicht. Da die Kapselung von Funktionalitäten entlang von Geschäftsobjekten gemäß dem Dienst-Konzept der SOA und eine Aggregation von verteilten Daten über die Portalanwendung im Sinne der Desktop Integration erfolgt, werden in dem AMD wesentliche Eigenschaften der SOA berücksichtigt (Anforderung 15 aus Abschnitt 6.4.2).

6.5.2 Gesamtarchitektur

In Abbildung 6.5-4 ist die Architektur des Prototyps für eine agentenbasierte elektronische Patientenakte zur Realisierung einer Virtualisierungsschicht oberhalb der Informationssystemlandschaft dargestellt, wobei dabei Rechnerknoten fokussiert werden. Diese Architektur wird im Folgenden beschrieben:

Im Zentrum befindet sich das im Abschnitt 6.5.1 beschriebene AMD, welches zur Laufzeit erzeugt wird und Daten von externen Informationsquellen aggregiert sowie temporär speichert. In der realisierten Umsetzung dieses Konzepts werden die folgenden Elemente angebunden:

- Krankenhausinformationssystem
- (Simuliertes) Arztpraxisinformationssystem
- Kartenlesegerät bzw. der für den Zugriff erforderliche Konnektor als Elemente der TI für die eGK
- Portallösung

In Abbildung 6.5-4 ist insbesondere die Verteilung der Systeme auf unterschiedliche Standorte angedeutet. Weiterhin wird die Aufteilung der Applikation auf stationäre und mobile Endgeräte dargestellt.

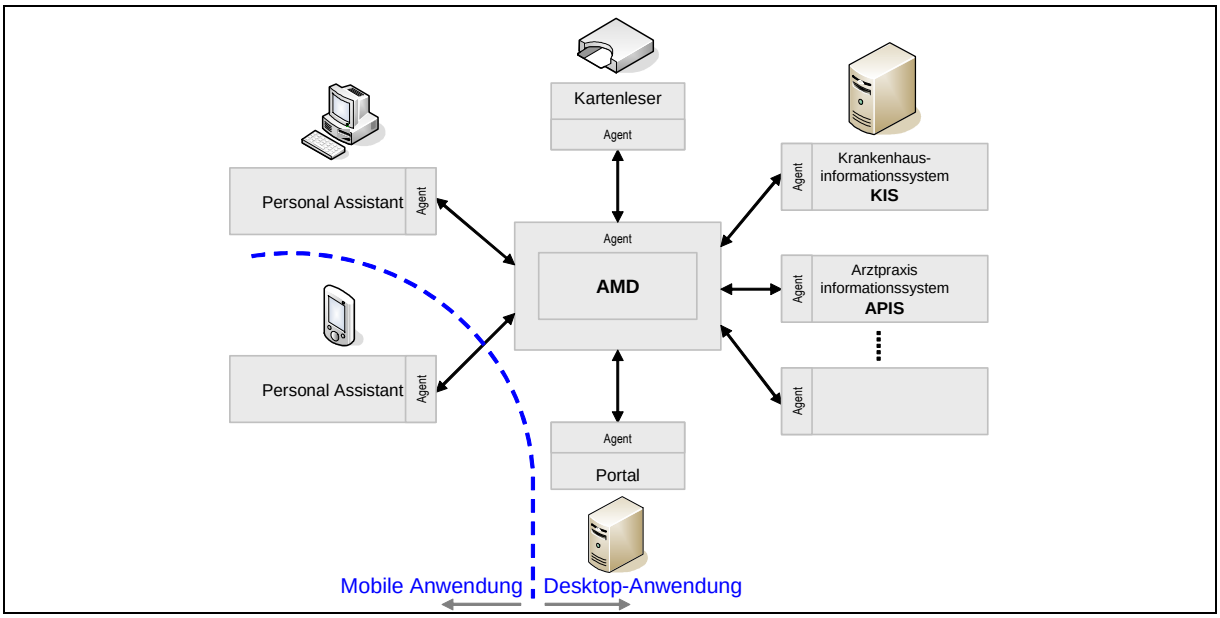

Abbildung 6.5-4: ***Schematische Darstellung der Anwendungsumgebung***
Quelle: *Hillebrand* (2006, 66) und in Anlehnung an *Schweiger et al.* (2006a, 37)

Nicht agentenbasierte Informationssysteme wie das gewählte KIS werden durch spezielle Wrapper-Agenten in Übereinstimmung mit der Vorgehensweise bei *Jennings* (2001, 40) in das Agentensystem integriert (siehe auch Abbildung 6.5-5). Diese Wrapper-Agenten kapseln die Funktionalität der nicht agentenbasierten Informationssysteme. Dazu werden ACL-Anfragen von anderen Software-Agenten, z.B. vom AMD, von den Wrapper-Agenten in die proprietären Funktionsaufrufe der jeweiligen Informationssysteme transformiert. Die Ergebnisse der Anfragen an die Informationssysteme werden wiederum vom Wrapper in eine Agentennachricht eingebettet, um vom ursprünglich anfragenden Agenten weiterverarbeitet werden zu können. In dem in Abbildung 6.5-5 dargestellten Beispiel sind die anfragenden Agenten diejenigen aus dem AMD.

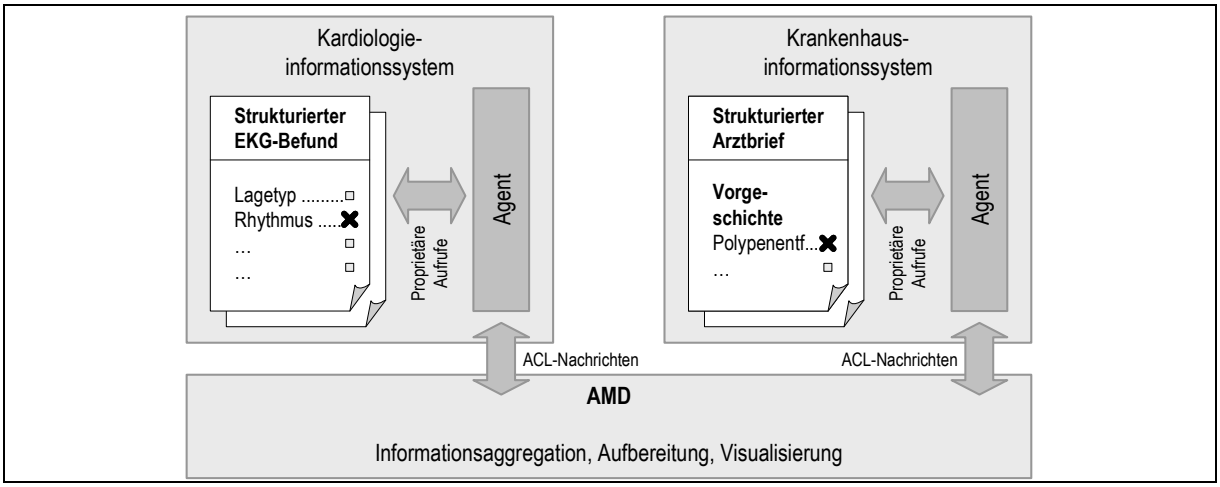

Abbildung 6.5-5: ***Integration von Informationssystemen in das Agentensystem***
Quelle: In Anlehnung an *Schweiger et al.* (2006b, 97)

Bei der in der vorliegenden Arbeit beschriebenen Implementierung handelt es sich um eine Realisierung mit solchen Eigenschaften, die sich aus dem Prototypenbau (siehe dazu Abschnitt 1.8) ergeben. Deshalb wird das AMD beispielhaft an die weit verbreiteten KISe SAP IS-H bzw. i.s.h.med (Krcmar et al. 2007, 24-25) in Teilen simuliert angebunden. Ein weiteres System zur Aggregation von Daten aus einem entfernten PVS wird über eine geeignete Komponente simuliert. Dabei werden Schnittstellen implementiert, die eine tatsächliche Anbindung eines PVSs an das Agentensystem vereinfachen.

6.6 Modellierung eines ausgewählten Szenarios aus der OP-Vorbereitung

Aus der umfangreichen Prozessbeschreibung für die Behandlung des kolorektalen Karzinoms (siehe Abschnitt 6.2.4) wird im Folgenden ein Ausschnitt gewählt, um zu demonstrieren, wie der Weg von der Analyse zu einer agentenbasierten Lösung beschritten werden kann. Dazu werden in einem ersten Schritt in einer Fallstudie (Bastian 2005) die OP-Vorbereitung, OP und Betreuung unmittelbar nach der OP innerhalb des genannten Behandlungsprozesses detailliert betrachtet. Die Fokussierung auf diesen Prozessausschnitt wurde gewählt, weil dabei insbesondere informationslogistische Anforderungen sowie die im Gesundheitswesen vorherrschende Dynamik deutlich werden. Der für die folgenden Ausführungen betrachtete Teilprozess umfasst die in Abbildung 6.2-5 dargestellten EPK-Funktionen „Aufklärung zur Anaesthesie und OP“, sowie „Anaesthesie“ und „OP“.

6.6.1 Überblick über den gewählten Prozessausschnitt

In der am Referenzklinikum München rechts der Isar der Technischen Universität München durchgeführten Fallstudie (Bastian 2005) werden u.a. die OP-Vorbereitung, die OP sowie die Betreuung unmittelbar nach der OP aus der Sicht der Anaesthesie detailliert analysiert. Die im Folgenden beschriebenen Schritte der genannten Teilprozesse sind zusammengefasst aus der Ausführung bei *Bastian* (2005, 68-73 und 81-82) und sind in Abbildung 6.6-1 dargestellt:

Entschied das Tumorboard auf Operation, wird parallel der OP-Plan erstellt und es werden zur Vorbereitung der Operation erforderliche Voruntersuchungen durchgeführt. Die Art der Untersuchung hängt vom Zustand des Patienten ab und kann z.B. Röntgenaufnahmen oder EKG-Untersuchungen umfassen. Als Ergebnis der durchgeführten Untersuchungen werden Befunde erstellt, die z.T. im KIS hinterlegt werden. Sobald die Voruntersuchungen durchgeführt sind, kann ein Termin für ein Gespräch mit dem Operateur vereinbart werden. Dabei wird vom Patienten ein Aufklärungsbogen über die Vorgehensweise und Risiken der OP unterschrieben.

Es folgt ein Anamnesegespräch[50] mit dem Anaesthesisten: Die dazu erforderlichen Informationen werden dem lokalen KIS, der digitalen und Papier basierten Patientenakte sowie dem zuvor vom Patienten ausgefüllten Aufklärungs- und Anamnesebogen entnommen. Zusätzlich

[50] Für die folgenden Ausführungen zur OP-Vorbereitung sind zwei unterschiedliche Bedeutungen des Begriffes Prämedikation zu differenzieren: Die Prämedikation im eigentlichen Sinne umfasst die Gabe der Medikation, üblicherweise 24 Stunden vor der OP. Im klinischen Sprachgebrauch ist der Begriff der Medikation auch im weiteren Sinne gebräuchlich: Dabei wird der gesamte Prozess mit dem Anamnesegespräch, der Befundsichtung, Verordnung der Medikation sowie der Gabe der Medikation verstanden.

wäre dabei die Bereitstellung von Daten aus der bisherigen Krankheitsgeschichte des Patienten wünschenswert. Diese Datenmenge müsste als Aggregation aus den Informationssystemen der beteiligten Leistungserbringer und somit als eine virtuelle Patientenakte (Roland Berger & Partner GmbH – International Management Consultants 1997, 10 und 36-39) zusammengestellt werden. Damit könnten z.B. bisher erfasste Befunde, Diagnosen, Therapien und Medikationen der gesamten Krankengeschichte des Patienten eingesehen und in die weitere Entscheidungsfindung integriert werden. Weil diese IT-Unterstützung in dem analysierten Teilprozess des Referenzklinikums nicht gegeben ist, wird als Schwerpunkt der vorliegenden Arbeit ein geeigneter Lösungsvorschlag beschrieben.

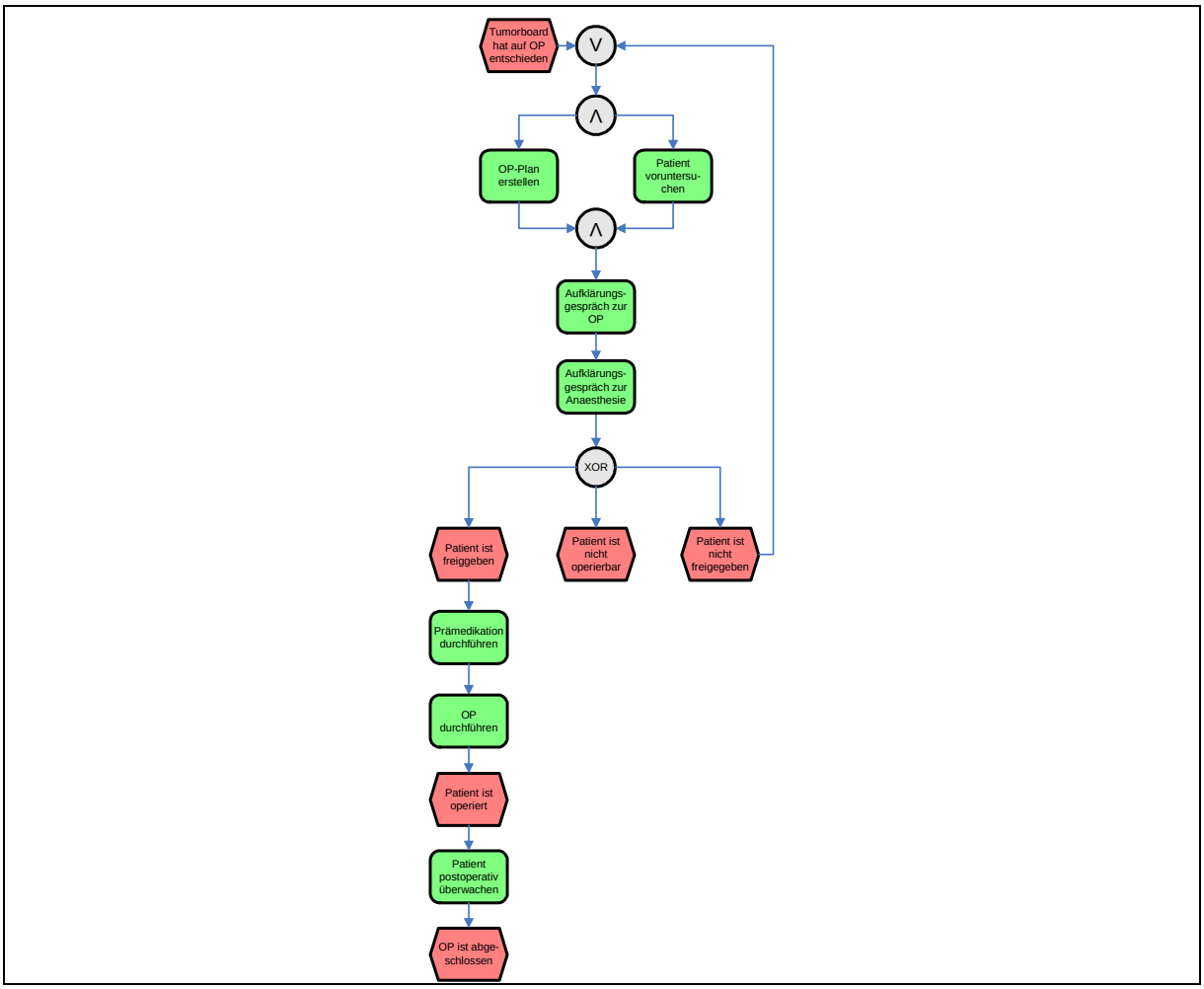

Abbildung 6.6-1: ***OP-Vorbereitung, OP und Betreuung des Patienten***
Quelle: Eigene Darstellung, in Anlehnung an *Bastian* (2005, 72)

Werden alle Voraussetzungen für eine problemlose Operation vom Patienten erfüllt, kann dieser für die geplante OP freigegeben werden. Es folgt die Aufklärung des Patienten über die Vorgehensweise und Risiken der Anaesthesie. Der Patient drückt seine Einwilligung dazu durch eine Unterschrift auf dem Aufklärungs- und Anamnesebogen aus. Das Ergebnis des gesamten Gesprächs wird in der Prämedikationsverordnung auf dem Prämedikationsbogen festgehalten. Vor der Operation, üblicherweise 24 Stunden zuvor, wird mit der Prämedikation (hier ist die Gabe der Medikation gemeint) begonnen. Nach der OP erfolgt abschließend eine

postoperative Überwachung des Patienten. Diese Überwachung wird ggf. im Intensivbereich durchgeführt.

Kann der Patient für die Operation hingegen nicht freigegeben werden, werden in Abhängigkeit von den im Aufklärungsgespräch zur Anaesthesie gewonnenen Ergebnissen weitere Voruntersuchungen durchgeführt oder auch ggf. ein neuer Termin für die Operation vereinbart. Der Anaesthesist kann evtl. auch zu der Entscheidung gelangen, dass der Patient nicht operierbar ist.

Aus der beschriebenen Darstellung des Prozessausschnittes wird ersichtlich, dass einerseits eine virtuelle Patientenakte für die Entscheidungsfindung des Anaesthesisten einen Mehrwert hinsichtlich der Belastbarkeit seiner Entscheidung leisten kann. Diese virtuelle Patientenakte erfordert insbesondere die Integration von Informationen des Patienten aus seiner bisherigen Krankheitsgeschichte.

Andererseits wird deutlich, dass ein Informationssystem flexibel auf sich dynamisch ändernde Rahmenbedingungen reagieren muss. Kann in dem beschriebenen Prozessausschnitt die Operation nicht erfolgen, muss der geplante Termin freigegeben sowie ein neuer gefunden werden. Da dafür jedoch üblicherweise umfangreiche manuelle Koordinationsarbeit erforderlich ist, kann ein Informationssystem mit entsprechender Funktionalität einen Mehrwert hinsichtlich Steigerung der Arbeitseffizienz leisten. Mit diesen Begründungen wird die Reduktion des gewählten Behandlungsprozesses auf den in Abbildung 6.6-1 dargestellten Teilprozess deutlich.

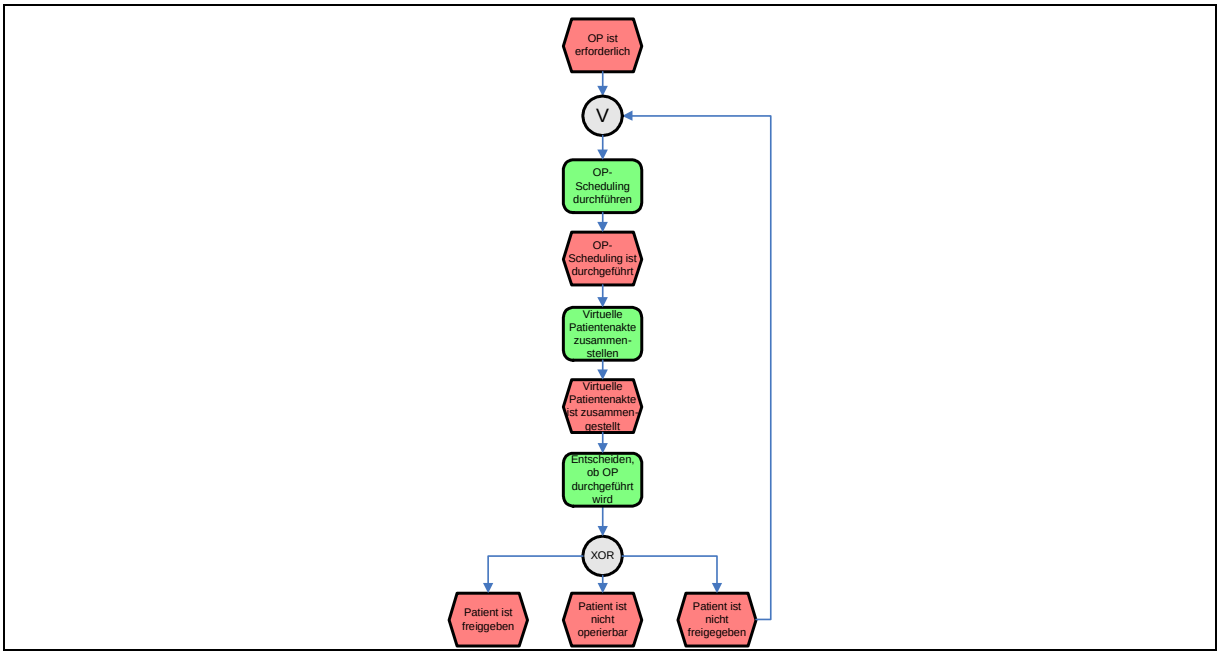

Abbildung 6.6-2: ***Modellierung des gewählten Prozessausschnittes der OP-Vorbereitung***
Quelle: Eigene Darstellung

Die für die Betrachtung des in der vorliegenden Arbeit beschriebenen Konzeptes relevanten Prozessteile des gewählten Prozessausschnittes sind in Abbildung 6.6-2 zusammengefasst.

Dabei werden die Funktionalitäten zur Aggregation der virtuellen Patientenakte sowie des dynamischen Schedulings entsprechend der Ausführungen weiter oben fokussiert.

6.6.2 Vorbemerkung zur Modellierung nach der ArBaCon-Methode

Es ist für die folgende Modellierung nach der ArBaCon-Methode anzumerken, dass sich die vorliegende Arbeit nicht mit der detaillierten Beschreibung jedes einzelnen Teilschrittes aus dieser Methode (siehe Abschnitt 1.7) befasst, sondern nur diejenigen Aspekte fokussiert, die für die Implementierung einer virtuellen Patientenakte auf einem Desktop-System für den Anwendungsbereich Informationslogistik im Klinikumsbetrieb relevant sind. Deshalb stehen in den Abschnitten 6.6.3 mit 6.6.6 die Ergebnisse der einzelnen Etappen im Vordergrund und weniger die jeweils verwendeten Methoden und detaillierten Teilschritte.

Weiterhin wird der Weg zur Erarbeitung der Ergebnisse für die Konstruktion der Scheduling-Anwendung, der mobilen Anwendung, der Portalanwendung sowie der Prozesssteuerung nicht ausführlich beschrieben, sondern es werden jeweils nur das Design und die Implementierung der genannten Teilanwendungen dargelegt. Die einzelnen Schritte nach der ArBaCon-Methode können auf diese Teilanwendungen aber übertragen werden und stellen somit keine Einschränkung hinsichtlich der gewählten Fokussierung der vorliegenden Arbeit dar.

6.6.3 Modellierung in der Etappe 1

In den folgenden Abschnitten 6.6.3.1 mit 6.6.3.5 werden ausgewählte Ergebnisse zur Modellierung der Domäne dargestellt. Grundlagen für diese Ausführungen sind die Beschreibung der Etappe 1 der ArBaCon-Methode bei *Reinke* (2003, 63-81) sowie die Anwendung dieser Methode bei *Bastian* (2005, 54-86), deren Ergebnisse in den genannten Abschnitten zusammengefasst und erweitert werden.

6.6.3.1 Modellierung von Anwendungsfällen

Die aus dem in Abbildung 6.6-1 dargestellten Prozess abgeleiteten Anwendungsfälle sind in Abbildung 6.6-3 dargestellt. Dabei erfolgt entsprechend den Ausführungen in Abschnitt 6.6.1 die Fokussierung auf die relevanten Aspekte der OP-Vorbereitung.

Die identifizierten Organisationseinheiten werden gemäß der ArBaCon-Modellierung (Reinke 2003, 66) durch ein Akteurssymbol mit einem ausgefüllten Kreis dargestellt. Akteure erhalten für ihre Namensgebung das Präfix „A“, Ressourcen werden durch das Präfix „R“ gekennzeichnet (Reinke 2003, 71).

Außer bei der Anamneseerhebung und Befundsichtung ist für den Zugriff auf Informationen lediglich das klinikinterne KIS beteiligt. Bei der Einsichtnahme in die virtuelle Patientenakte ist der Zugriff auf relevante Informationssysteme beinhaltet, durch den Informationen der bisherigen Krankheitsgeschichte bereitgestellt werden sollen. Dabei wird in dem modellierten Szenario auf Patientendaten aus dem PVS eines niedergelassenen Arztes zugegriffen. In der vorliegenden Arbeit wird vorgeschlagen, diesen Zugriff über die Kommunikationsinfra-

struktur der TI für die eGK zu realisieren. Eine Begründung dafür ist in Abschnitt 6.4.1 beschrieben.

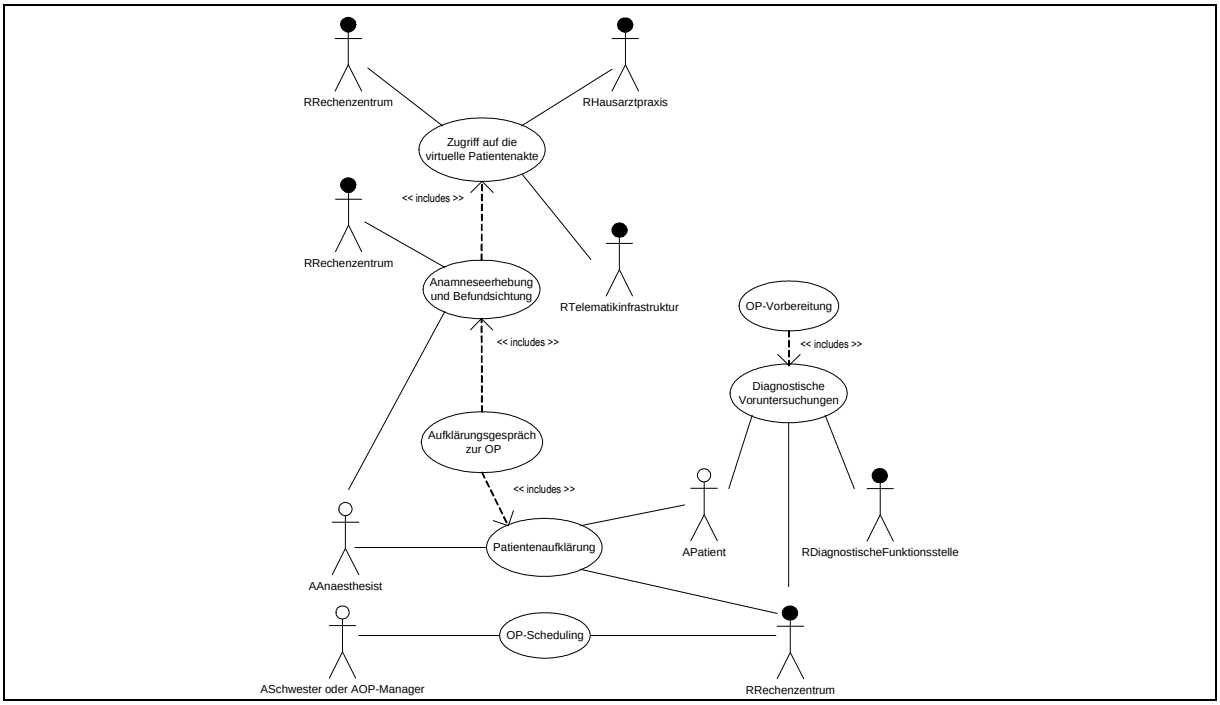

Abbildung 6.6-3: ***Anwendungsfälle zu dem gewählten Teilprozess der OP-Vorbereitung***
Quelle: Eigene Darstellung, in Anlehnung an *Bastian* (2005, 68 und 82)

Das OP-Scheduling wird im analysierten Anwendungsfall bisher manuell vorgenommen, besitzt aber Automatisierungspotenzial. Dazu wird in der vorliegenden Arbeit ein geeigneter Lösungsansatz dargestellt.

Der Anwendungsfall „Zugriff auf die virtuelle Patientenakte“ (Abbildung 6.6-3) zur Bereitstellung von Daten aus der virtuellen Patientenakte (siehe Abbildung 6.6-4) dient als Grundlage für die Entwicklung der Teilanwendung des MASs zur Informationsaggregation. Dieser Anwendungsfall wird gewählt, weil hier die Integration unterschiedlicher Informationssysteme sinnvoll ist und damit das Potenzial von Software-Agenten für die Informations- und Prozessintegration entsprechend der EAI bzw. SOA demonstriert werden kann.

Beim Zugriff auf Daten über die TI ist der dort vorgegebene Konnektor involviert. Der Zugriff auf die virtuelle Patientenakte kann z.B. über eine auf der eGK gespeicherte Verweisliste realisiert werden (Aktionsforum für Telematik im Gesundheitswesen der GVG (Gesellschaft für Versicherungswissenschaft und -gestaltung e.V.) 2004, 27-29 und 40-41). Alternativ könnte diese Verweisliste auch auf einem zentralen Server der TI gespeichert werden. Dabei ist insbesondere vorteilhaft, dass der Speicherplatz erweiterbar ist. Dies ist bei der eGK nicht gegeben. Weil der Zugriff auf diese Verweislisten durch den Konnektor mediiert wird, wird diese Funktionalität vor den Applikationen verborgen. Insofern hat die Implementierungsentscheidung über den Speicherort der Verweisliste keinen wesentlichen Einfluss auf die Qualität der prototypischen Implementierung. Zur Reduktion der Komplexität wird für die vorliegende Implementierung die Alternative mit der Speicherung der Verweisliste auf der eGK ge-

wählt. Die dort hinterlegten und vom Patienten für die Einsicht durch Leistungsträger freigegebenen Verweise enthalten die Daten zu in relevanten Informationssystemen gespeicherten Informationen. Um auf Verweislisten auf der eGK zuzugreifen, werden die eGK und der HBA des Leistungserbringers benötigt. Der Zugriff auf die Daten der Karten erfolgt über den Konnektor.

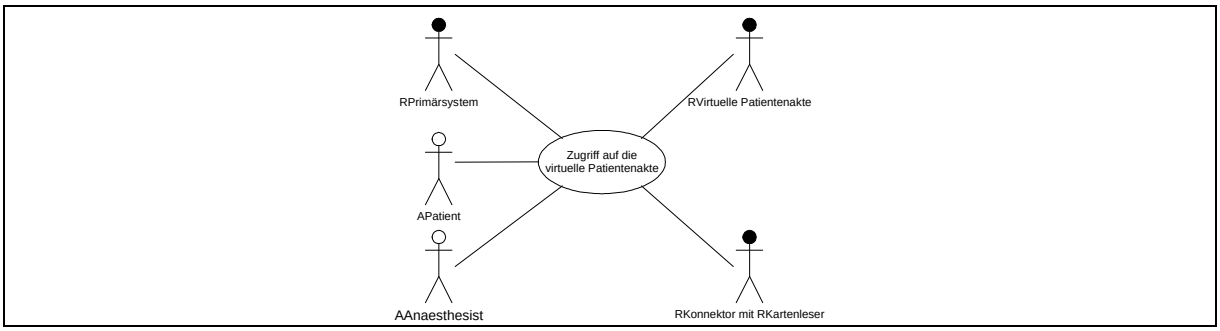

Abbildung 6.6-4: ***Anwendungsfall Zugriff auf die virtuelle Patientenakte***
Quelle: Eigene Darstellung

Externe Anfragen an das Primärsystem werden ebenso über den Konnektor an dieses weitergeleitet. In der virtuellen Patientenakte werden schließlich alle aggregierten Informationen visualisiert.

6.6.3.2 Modellierung von Organisationsstrukturen

Die analysierte Organisationsstruktur ist in Abbildung 6.6-5 in der von *Reinke* (2003, 67) vorgeschlagenen Notation dargestellt und wird im Folgenden beschrieben:

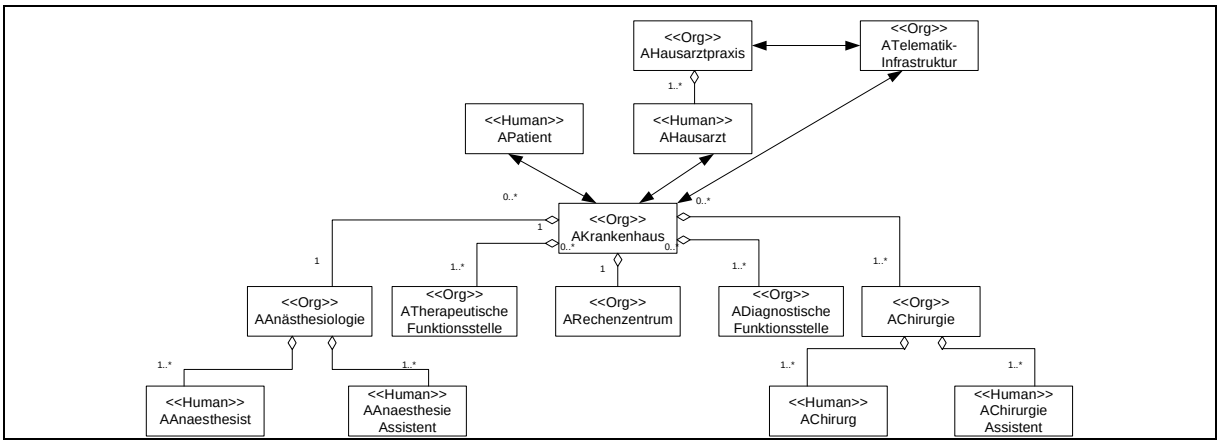

Abbildung 6.6-5: ***Organisationsstruktur***
Quelle: In Anlehnung an *Bastian* (2005, 63)

In einer Hausarztpraxis sind mindestens einer oder mehrere Hausärzte aggregiert. Zwischen einem Krankenhaus und einem Patienten bzw. einem Hausarzt existieren jeweils Bekanntschaftsbeziehungen, dargestellt durch Doppelpfeile. Diese Beziehungen sind wichtig, weil nur Agenten, die im Auftrag ihrer jeweiligen Akteure handeln, Interaktionen bilden können, wenn sie in der Organisationsstruktur durch eine Relation in Beziehung stehen (Reinke 2003, 67).

Zwischen dem betrachteten Referenzklinikum und den niedergelassenen Ärzten existiert eine ausgeprägte Kooperation. Patientendaten werden bisher jedoch nicht digital ausgetauscht. In dem gewählten Ausschnitt aus dem Krankenhaus bilden das Rechenzentrum, diagnostische sowie therapeutische Funktionsstellen, die Anaesthesiologie und die Chirurgie relevante Organisationseinheiten. Das Rechenzentrum stellt die basale Infrastruktur für den digitalen Datenaustausch zur Verfügung. Den Abteilungen für die Anaesthesiologie und Chirurgie sind die Akteure AAnaesthesist und AAnaesthesieassistent bzw. AChirurg und AChirurgieassistent zugeordnet.

Um den Datenaustausch zwischen mehreren Institutionen und Sektoren zu ermöglichen, stellt die TI die erforderlichen Mechanismen zur Verfügung. Dies wird in der Organisationsstruktur durch die Bekanntschaftsbeziehungen zwischen Praxis und TI bzw. Krankenhaus und TI modelliert.

6.6.3.3 Lokationsmodell

Die Grundlage für die spätere Allokation der einzelnen Teilsysteme bildet das Lokationsmodell[51] (Reinke 2003, 67-68). Der für die Implementierung des Prototyps zur Informationsaggregation aus verteilten Informationssystemen relevante Modellausschnitt ist in Abbildung 6.6-6 dargestellt und wird im Folgenden erläutert:

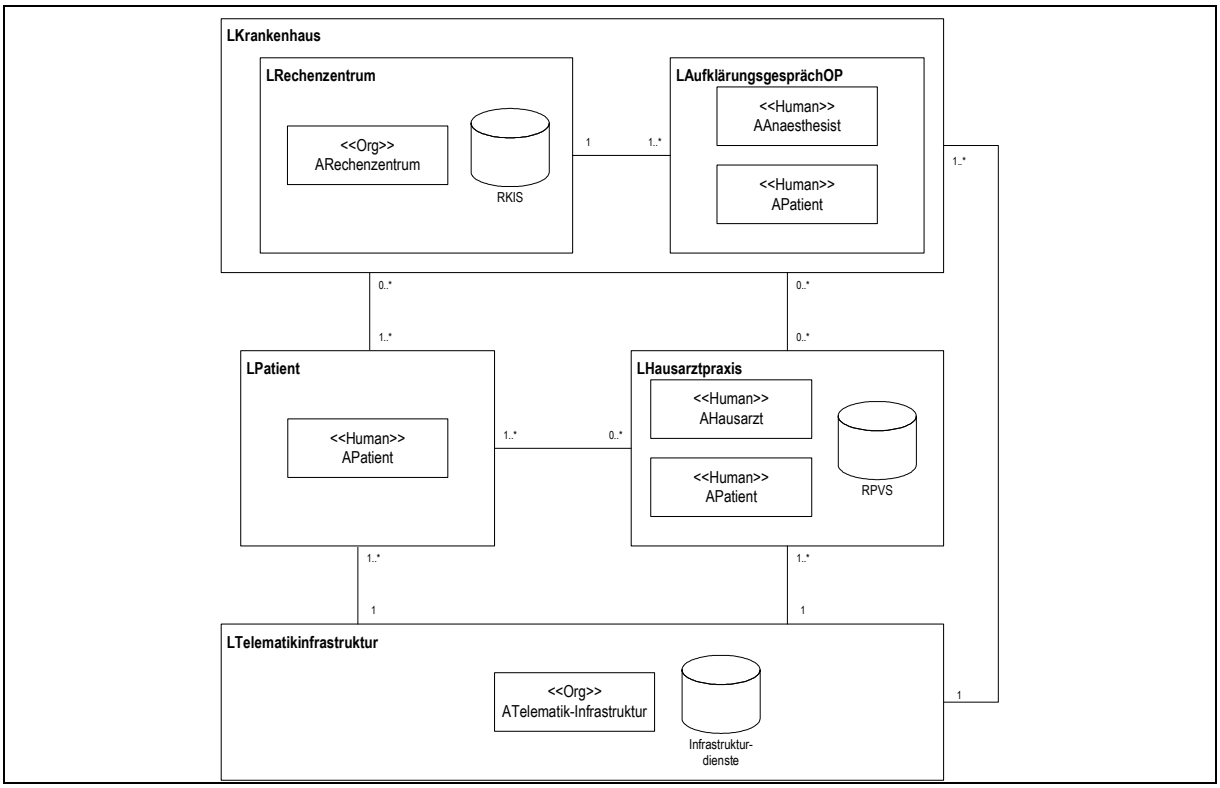

Abbildung 6.6-6: ***Lokationsmodell***
Quelle: In Anlehnung an *Bastian* (2005, 64)

[51] Lokationen werden mit dem Präfix „L“ gekennzeichnet.

Patientendaten sind in dem betrachteten Szenario auf zwei Speicherorte verteilt: Die Funktionalität des KISs, bezeichnet mit RKIS, wird vom Rechenzentrum zur Verfügung gestellt. Das Praxisverwaltungssystem RPVS wird mit der Hausarztpraxis assoziiert. Um die aktive Rolle des Patienten zu verdeutlichen, ist dieser einem eigenen Bereich zugeordnet (Bastian 2005, 65). Je nach Status im Behandlungsprozess kann der Patient anderen Lokationen zugeordnet werden, womit die Idee der aktiven dynamischen Behandlungsräume (Schweiger/Krcmar 2004) geeignet berücksichtigt werden kann. Ein weiterer relevanter Bereich ist die Prämedikation. Die TI mit ihren Diensten ist als separate Lokation modelliert.

Die in Abbildung 6.6-6 dargestellten Kardinalitäten werden im Folgenden beschrieben: Die Bereiche innerhalb eines Klinikums greifen auf die Dienste eines zentralen Rechenzentrums zu. Umgekehrt werden diese Dienste von mehreren Einrichtungen in Anspruch genommen. Deshalb werden die Kardinalitäten dieser Assoziationen mit einer Beziehung (1, 1..*) dargestellt. Die TI bietet ihre Dienste allen Gesundheitseinrichtungen an. Umgekehrt werden mit der Einführung der eGK alle Einrichtungen an diese Infrastruktur angebunden. Deshalb erhält die Assoziation zwischen medizinischen Einrichtungen und der TI die Kardinalität (1, 1..*). Ein Krankenhaus bzw. eine Hausarztpraxis behandeln mindestens einen Patienten, umgekehrt lässt sich ein Patient nicht notwendigerweise, aber möglicherweise von mehreren Krankenhäusern oder Hausärzten behandeln. Deshalb erhalten diese Beziehungen die Kardinalität (0..*, 1..*). Niedergelassene Ärzte und Krankenhäuser können miteinander kooperieren, die Kardinalität ist dabei in beiden Richtungen nicht begrenzt. Daher besitzt diese Assoziation die Kardinalität (0..*, 0..*).

6.6.3.4 Modellierung von Geschäftsobjekten

Als Grundlage für die Erstellung von Geschäftsmodellen werden zunächst vorhandene Informationssysteme und ihre Schnittstellen sowie die für die Patientenakte verwendeten Medien beschrieben.

Der Überblick über sämtliche im Referenzklinikum eingesetzten Informationssysteme ist in Abbildung 6.6-7 dargestellt. Im Zentrum der Informationssystemlandschaft steht ein Kommunikationsserver mit den Aufgaben der Gewährleistung des Nachrichtenaustausches zwischen den angebundenen Systemen und ggf. der Transformation von Daten mit unterschiedlichen Formaten (siehe dazu die Ausführungen zu Kommunikationsservern in Abschnitt 4.6.1). Im Referenzklinikum werden über den Kommunikationsserver zwischen den Informationssystemen lediglich Patientenstamm- und Bewegungsdaten ausgetauscht (Bastian 2005, 76).

Neben den in Abbildung 6.6-7 dargestellten heterogenen Subsystemen nimmt das System SAP R/3® eine zentrale Stellung ein. Damit werden administrative Verfahren unterstützt. Der IS-H-Teil dieses Systems unterstützt speziell die Belange der Patientenverwaltung. Durch das i.s.h.med-Modul werden medizinische Prozesse unterstützt. Weil über den Kommunikationsserver vom SAP-System auch andere Informationssysteme angesprochen werden können und das SAP-System mit seinen Erweiterungen IS-H und i.s.h.med im MRI das führende KIS darstellt, ist eine Ankopplung der virtuellen Patientenakte an das SAP-System in der prototypischen Implementierung ausreichend.

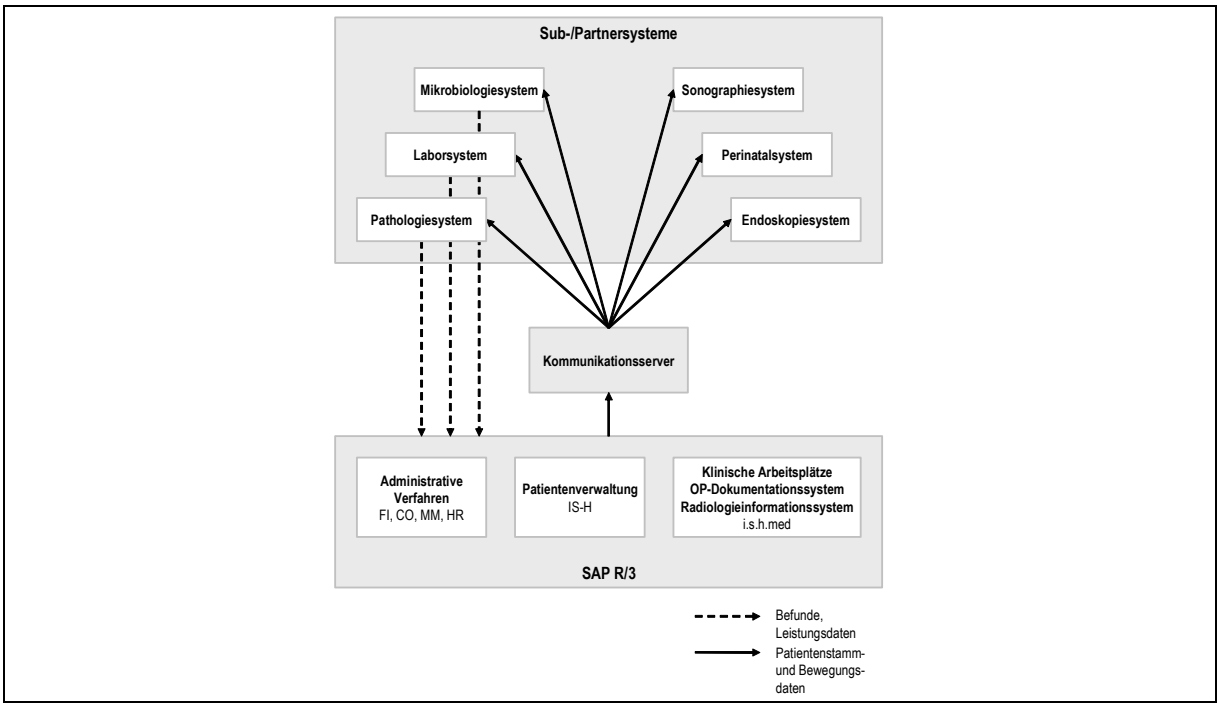

Abbildung 6.6-7: ***Informationssysteme im Klinikum MRI***
Quelle: In Anlehnung an *Rechenzentrum des Klinikum rechts der Isar der TU München* (2003, 28)

Eine Extraktion von Daten aus dem SAP-System ist prinzipiell durch folgende Schnittstellen möglich (Bastian 2005, 78):

- HL7-Schnittstelle, Version 2.1
- Proprietäre HCM-Schnittstelle, Versionen 1.1 und 1.2
- Business Application Programming Interfaces (BAPI)

Für eine Auswahl einer geeigneten Schnittstelle zur Berücksichtigung in der prototypischen Implementierung der vorliegenden Arbeit sind die folgenden Ausführungen relevant: Die HL7-Version 2.1 ist nicht mehr auf dem aktuellen Stand (siehe dazu die Ausführungen in Abschnitt 4.11.2.1.1) und unterstützt nicht die für die automatisierte Weiterverarbeitung von Daten sinnvolle XML-Kodierung. Die proprietäre HCM-Schnittstelle besitzt den Nachteil der beschränkten Erweiterbarkeit. Als die für das angestrebte Ziel der Informationsintegration am besten geeignete Schnittstelle ist der Einsatz von Business Objects[52] (BO) des SAP-Systems, auf die über BAPIs[53] zugegriffen werden kann. Nicht unterstützte BAPI-Funktionalitäten können durch eigene ABAP-Programme[54] (Advanced Business Application Programming) ergänzt werden. Damit können die jeweils relevanten Daten aus dem führenden KIS in eine an-

[52] Business Objects werden als Komponenten definiert, welche relevante Daten und Objekte aus dem SAP-System nach ausgewählten Kriterien entsprechend der Objektorientierung strukturieren (2006). Weitere Informationen zu SAP Business Objects können auf den folgenden Seiten abgerufen werden: http://help.sap.com, zugegriffen am 26.02.2007.

[53] Eine BAPI realisiert eine wohl definierte Programmierschnittstelle für den Zugriff auf die gekapselten Daten von Business Objects und realisiert somit die Voraussetzung für die Integration von SAP-Daten in ein anderes Anwendungssystem.

[54] ABAP ist auf den folgenden Seiten ausführlich dargestellt: http://help.sap.com, zugegriffen am 26.02.2007.

dere Anwendung, z.B. in die virtuelle Patientenakte der vorliegenden Arbeit, exportiert werden. Für die Integration von BAPI-Aufrufen innerhalb einer externen Anwendung sind spezielle Bibliotheken (z.B. JCo[55], siehe Abschnitt 6.4.1) erforderlich.

Mit den existierenden BAPIs zur Patientenverwaltung können aus IS-H lediglich Patientenstammdaten, Verwaltungsdaten, Prozeduren, Diagnosen und Falldaten ausgelesen werden (Bastian 2005, 78). Im BO MedicalDocument des i.s.h.med-Systems werden medizinische Dokumente wie Befunde zu einem Patienten hinterlegt (Bastian 2005, 79). Diese Dokumente werden nicht in der dem SAP-System zugrunde liegenden Datenbank abgelegt, sondern werden über einen Link, der den Speicherort der Daten (Dateisystem, SAP-Archiv, SAP-Dokumentenmanagementsystem) angibt, in das System integriert (Bastian 2005, 79). Über die BAPI des BO MedicalDocument wird lediglich die Manipulation der Verweisangaben hinsichtlich des Anlegens neuer Verweise oder der Änderung von deren Status unterstützt (Bastian 2005, 79). Somit können mit den vorhandenen BAPIs keine vollständigen Dokumente, sondern nur bestimmte Teile daraus extrahiert werden. Deshalb werden in der vorliegenden Arbeit Teile der Anbindung an das KIS entsprechend dem Prototypenbau simuliert, indem medizinische Dokumente im Dateisystem abgelegt werden und über einen simulierten BAPI-Aufruf der virtuellen Patientenakte zur Verfügung gestellt werden.

In der beobachteten Informationssystemlandschaft des Klinikums werden keine IT-Systeme zur Unterstützung einer standardisierten Behandlung nach Patientenpfaden (Clinical Pathway, CPW, Hellmann 2002) identifiziert (Bastian 2005, 77). Somit besteht hier Potenzial für eine geeignete Prozessunterstützung (siehe dazu auch die Abschnitte 6.7.6 und 6.8.7).

In der Fallstudie konnte von *Bastian* (2005) im Referenzklinikum eine hybride Patientenakte beobachtet werden, die sich aus digitalen Einträgen und Papier basierten Dokumenten zusammensetzt. Die Papier basierte Akte enthält nicht nur Dokumente, die innerhalb des Klinikums erzeugt wurden, sondern auch vom Patienten mitgebrachte Dokumente wie Arztbriefe (Bastian 2005, 79). Die in der Fallstudie für den beschriebenen Prozess als relevant identifizierten Dokumente und das jeweilige Medium werden in Tabelle 6.6-1 zusammengefasst. Insofern stehen Informationen zwar Institutionen übergreifend zur Verfügung, allerdings auf unterschiedlichen Medien. Hier lässt sich deutlich der Mehrwert einer virtuellen, institutionsübergreifenden, elektronischen Patientenakte nachvollziehen.

[55] Auf der Basis von JCo wird bei *Kikic/Ratkovic* (2006) eine API entwickelt, die den Zugriff auf die SAP-Informationen entsprechend der Objektorientierung vereinfacht. Damit wird dem Entwickler ermöglicht, von den spezifischen Eigenschaften der Bibliotheken zur Anbindung von externen Applikationen an BAPI-Aufrufe zu abstrahieren und damit den Entwicklungsprozess zur Ankopplung von weiteren Anwendungen zu vereinfachen..

Material	Bestandteile	Medium
Papier basierte Patientenakte	• Befunde • Diagnosen • Medikationen • Anamnese • Entlassbriefe • Arztbriefe	Papier
Prämedikationsbogen	• Zusammenfassung der Aufklärungsgespräche • Prämedikationsverordnung	Papier
Aufklärungs- und Anamnesebogen	• Aufklärung des Patienten • Einverständniserklärung des Patienten • Anamnesedaten	Papier
KIS	• Stammdaten • Elektronisch erfasste und gespeicherte Patientendaten wie Prozeduren, Diagnosen und medizinische Dokumente	Digital

Tabelle 6.6-1: ***Überblick über Bestandteile der hybriden Patientenakte***
Quelle: In Anlehnung *Bastian* (2005, 80-81)

Weil für eine erste prototypische Annäherung mit vertikaler Funktionalität die Aggregation von ausgewählten Dokumenten ausreichend ist, ist lediglich der für die Implementierung relevante Ausschnitt des Geschäftsobjektmodells der virtuellen Patientenakte in Abbildung 6.6-8 dargestellt.

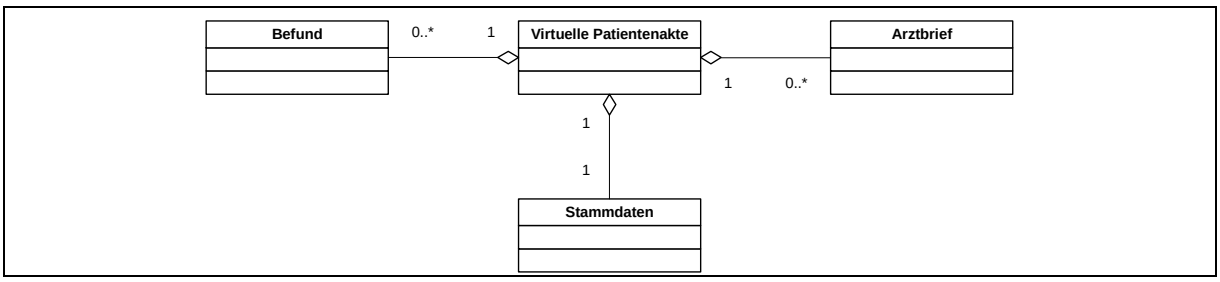

Abbildung 6.6-8: ***Geschäftsobjektmodell für die virtuelle Patientenakte***
Quelle: Eigene Darstellung

Für die prototypische Implementierung werden Befunde, Stammdaten und Entlassbriefe ausgewählt. Die daraus in der virtuellen Patientenakte aggregierten Daten können durch weitere Informationen aus anderen Informationssystemen ergänzt werden. Dazu ist die Implementierung entsprechend zu erweitern. Diese Ergänzung kann vorgenommen werden, weil die Architektur modular angelegt ist.

6.6.3.5 Prozessmodell

Für das ausgewählte Szenario zur Zusammenstellung der virtuellen Patientenakte werden im Folgenden die dafür relevanten Prozesse modelliert, durch die die Abarbeitungsweise von geeigneten Agenten beschrieben wird. Die dafür gewählten Aktivitätsdiagramme folgen der bei *Reinke* (2003) vorgeschlagenen, erweiterten UML-Notation für Aktivitätsdiagramme, um insbesondere agentenspezifische Eigenschaften geeignet abbilden zu können. Grau hinterlegte

Aktivitäten bezeichnen in einem solchen Diagramm Interaktionen im Gegensatz zu regulären Aktionen. Komposite Aktivitäten werden durch ein Dekompositionssymbol gekennzeichnet.

Grundlage für die Prozessmodellierung sind die in der bisherigen Modellierung erfassten Anwendungsfälle. In der vorliegenden Arbeit wird der Anwendungsfall „Zugriff auf die virtuelle Patientenakte“ aus Abschnitt 6.6.3.1 ausgewählt. Der Prozess zum Zugriff auf die virtuelle Patientenakte kapselt die beiden Teilprozesse zur Anfrage von Daten aus der TI und zur Zusammenstellung der eigentlichen Patientenakte. Diese Teilprozesse sind in Abbildung 6.6-9 dargestellt: Der Prozess wird vom Akteur AAnaesthesist initiiert. Zur Vorbereitung der Einsichtnahme in die Daten der virtuellen Patientenakte wird eine geeignete Anfrage an das AMD gestellt. Über die TI werden unter Verwendung der Verweisliste aus der eGK Daten aus lokal verteilten Informationssystemen, im Beispielszenario aus einem Krankenhaus (RKIS) und einer Praxis eines niedergelassenen Arztes (RPVS) abgefragt. Sobald die Daten eingetroffen sind, können sie vom Anwender begutachtet werden.

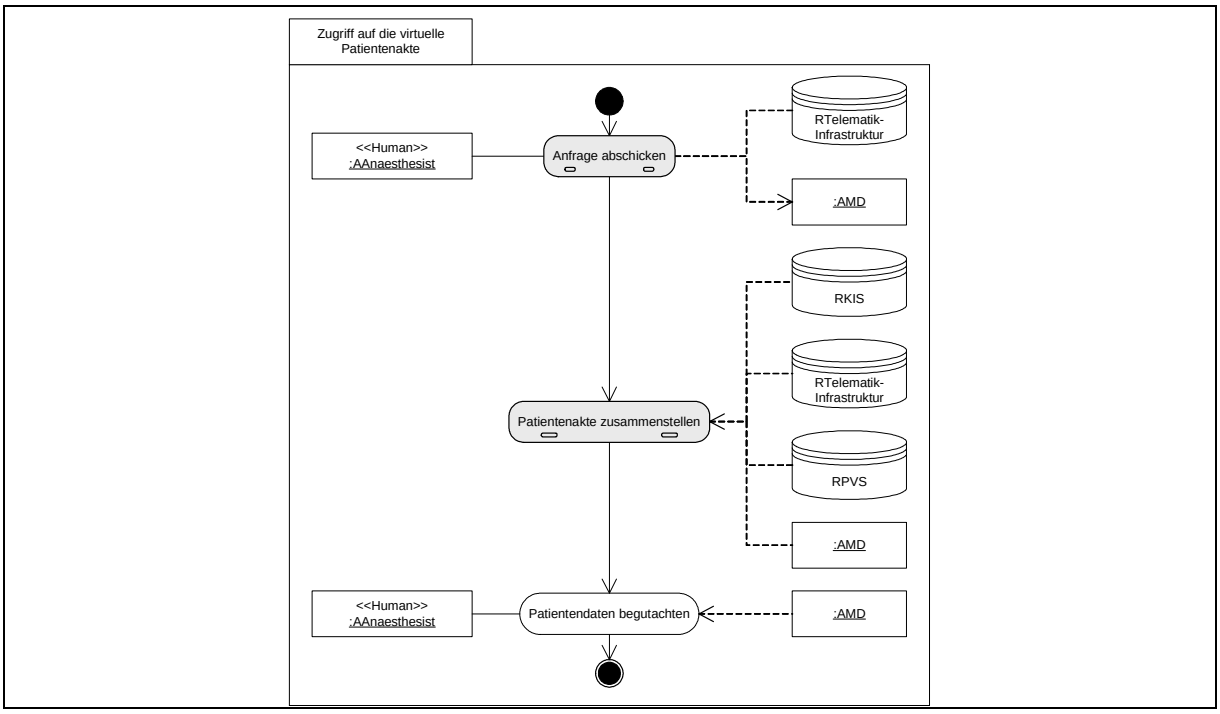

Abbildung 6.6-9: ***Prozess Zugriff auf die virtuelle Patientenakte***
Quelle: Eigene Darstellung, in Anlehnung an *Bastian* (2005, 84)

In Abbildung 6.6-10 ist die verfeinerte Prozessdarstellung zum Abschicken der Anfrage dargestellt. Im ersten Schritt wird die Patientenakte geöffnet. In der vorliegenden Implementierung ist der Zugriff auf die verteilt vorliegende Information der Patientendaten über eine Verweisliste auf der eGK realisiert. Um auf diese Informationen zugreifen zu können, ist eine Abfrage des AMD über das Kartenlesegerät erforderlich. Dieser Zugriff wird, entsprechend den Vorgaben aus der TI durch den jeweiligen Konnektor mediiert. Vor dem Auslesen der Daten wird überprüft, ob der Zugriff erlaubt ist. Die aus der eingelegten eGK ausgelesenen Daten werden über den Konnektor an das AMD weitergeleitet. An dieser Stelle wird deutlich, dass der tatsächliche Speicherort der Verweisliste durch den Zugriff über den Konnektor vor der Anwendung verborgen wird.

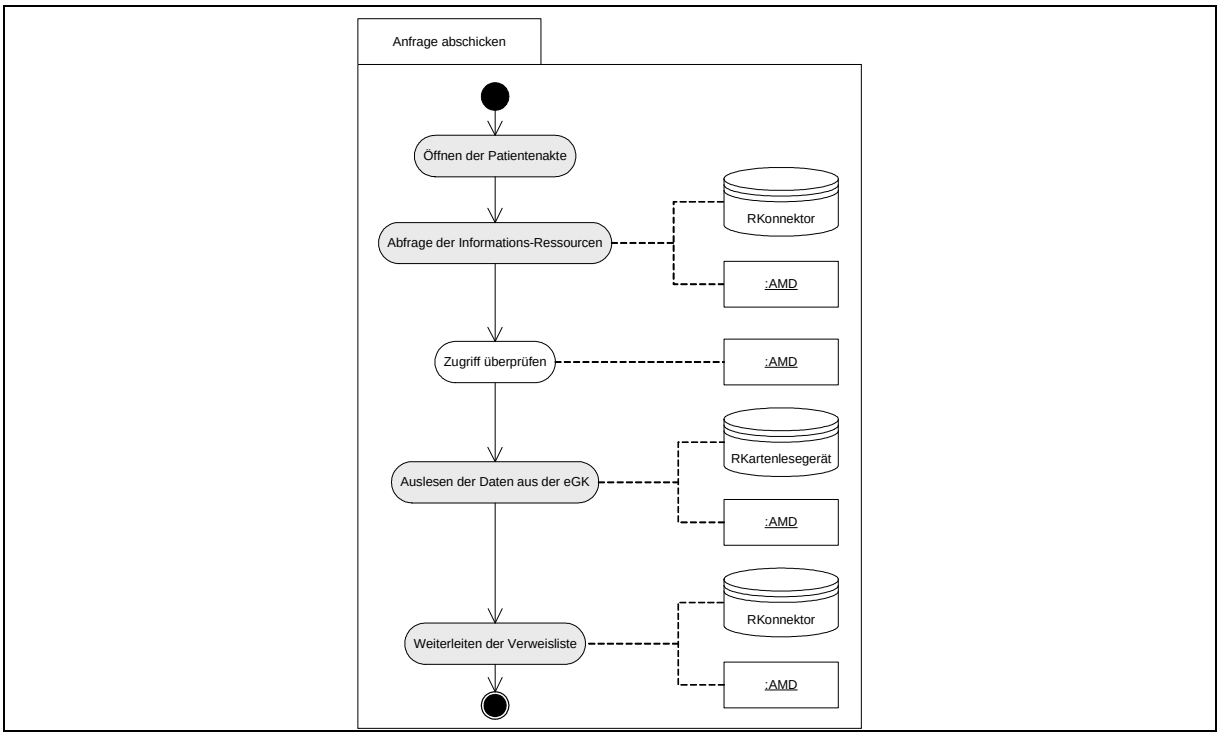

Abbildung 6.6-10: ***Prozess Anfrage abschicken***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Der Teilprozess zur Zusammenstellung der Daten für die virtuelle Patientenakte ist in Abbildung 6.6-11 dargestellt. Wesentliche Voraussetzung dabei ist die Berücksichtigung der in der eGK gespeicherten Verweise auf Informationssysteme mit bisherigen Patientendaten. Die Menge dieser Ressourcen wurde bereits im Teilprozess „Anfrage abschicken“ ermittelt. Es folgt die Planung und Steuerung der Abfrage der Patientendaten. Diese Abfrage erfolgt nebenläufig, weil Anfragen an die unterschiedlichen Informationssysteme parallel erfolgen können. Dabei wird zwischen einer Anfrage des aus Sicht des Anwenders lokalen bzw. entfernten Anwendungssystems differenziert. Die jeweiligen Abfragen werden so oft ausgeführt, wie Verweise auf der Karte gespeichert sind. Diese nebenläufige Ausführung ist entsprechend in Abbildung 6.6-11 gekennzeichnet.

In der parallelen Abarbeitung wird zunächst die Suchanfrage empfangen und anschließend an die identifizierten Informationssysteme bzw. die sie kapselnden Agenten versendet. Bei externen Suchanfragen erfolgt zusätzlich eine Weiterleitung der Anfrage an das entfernte Informationssystem. In beiden Fällen, lokale und entfernte Abfrage, werden dort die Suchanfrage ausgeführt und die gewünschten Daten aus dem Informationssystem entnommen. Das Ergebnis der Suchanfrage wird an das AMD weitergeleitet. Dieses wird von der virtuellen Patientenakte entgegengenommen und an der Benutzeroberfläche visualisiert. Sobald alle Suchanfragen wieder eingetroffen sind und für die Anzeige an der Benutzeroberfläche aufbereitet sind, endet die nebenläufige Bearbeitung. Der Prozess kann nun an der entsprechenden Stelle im Teilprozess „Zugriff auf die virtuelle Patientenakte“ fortgesetzt werden.

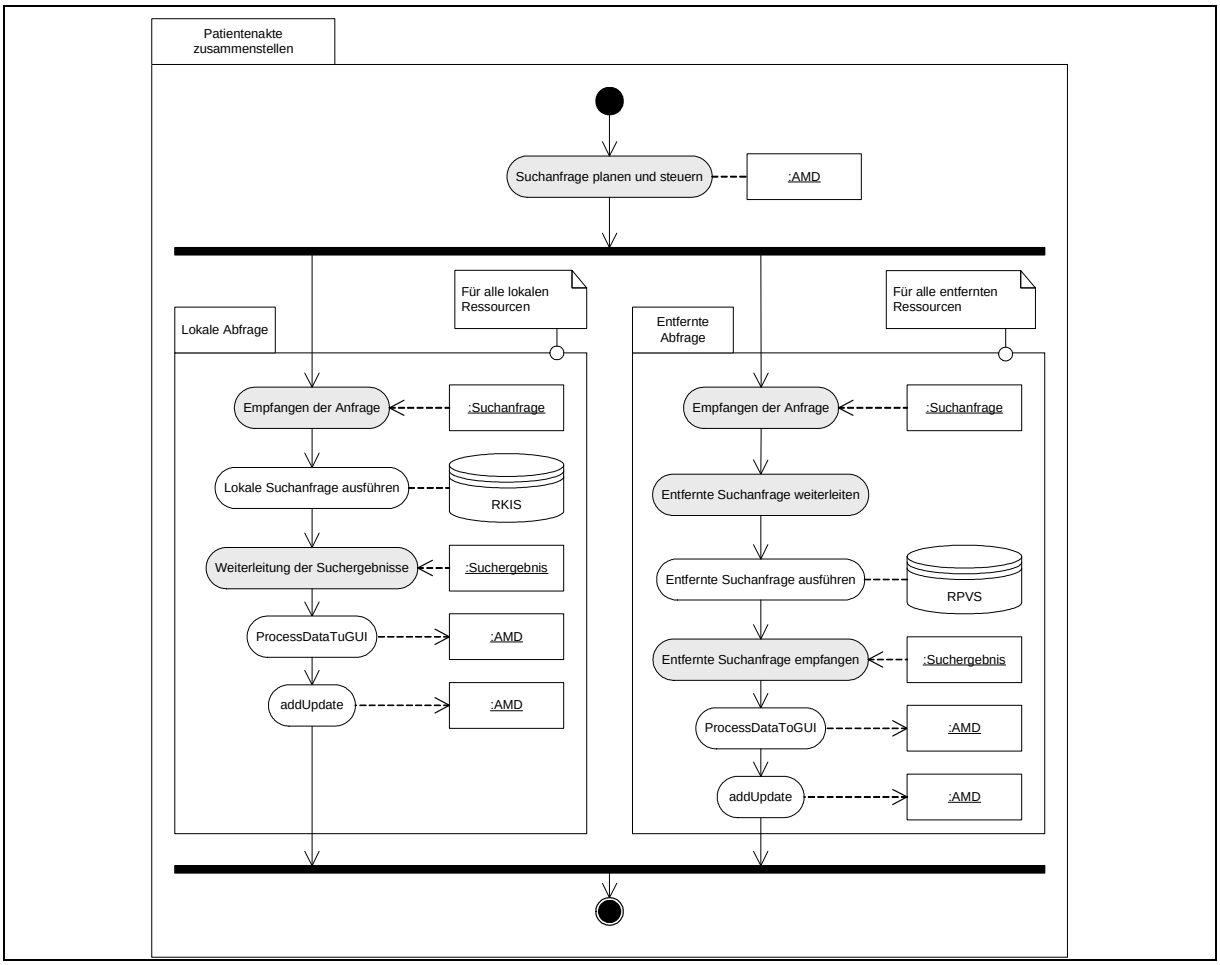

Abbildung 6.6-11: ***Prozess Zusammenstellung der virtuellen Patientenakte***
Quelle: Eigene Darstellung, in Anlehnung an *Bastian* (2005, 86) und *Reinke* (2003, 79)

6.6.4 Modellierung in der Etappe 2

Grundlagen für die Ausführungen in diesem Abschnitt sind die Beschreibungen bei *Reinke* (2003, 83-106) zur Etappe 2 der ArBaCon-Methode sowie bei *Bastian* (2005, 89-97) zur exemplarischen Anwendung der ArBaCon-Methode. Die Ergebnisse der Anwendung der Methode bei *Bastian* (2005, 89-97) werden in diesem Abschnitt zusammengefasst.

6.6.4.1 Abbildung von Akteuren, Material und Umgebungen

Entsprechend dem Vorgehensmodell bei *Reinke* (2003, 92) werden Akteure, Ressourcen und Umgebungen aus der Domänenmodellierung (siehe dazu Abschnitt 6.6.2) auf Autoritäten, Ressourcen und Agentenplattformen, dargestellt als Architekturmodelle, abgebildet. In diesem Transformationsschritt werden Domänenmodelle in Konstrukte der Software-Technik abgebildet. Für den vorliegenden Anwendungsfall ergeben sich die in Tabelle 6.6-2 dargestellten Abbildungen. Dabei ist zu beachten, dass die Entitäten aus der Domäne deutsche Bezeichnungen erhalten, während Architekturmodelle mit englischen Begriffen belegt werden (siehe Reinke 2003, 92), um die Bereiche der Domänen- und Architekturmodellierung zu differenzieren.

Actor	**Authority**
AHausarzt	APhysician
AAnaesthesist	AAnaesthesiologist
APatient	APatient
Material	**Resource**
RKIS	HIS
RPVS	GPIS
Location	**AgentPlatform**
LKrankenhaus	Hospital
LHausarztpraxis	General Practitioner
LTelematikinfrastruktur	Telematics

Tabelle 6.6-2: ***Abbildung von Akteuren, Ressourcen und Lokationen***
Quelle: In Anlehnung *Bastian* (2005, 89-90)

Es ist weiterhin anzumerken, dass die Umgebung, in der das Aufklärungsgespräch zur OP zwar in Abbildung 6.6-6 als separate Lokation modelliert ist, aus Gründen der Vereinfachung aber in der folgenden Betrachtung nicht mehr berücksichtigt wird. Vielmehr werden die in der Lokation LPrämedikation angesiedelten Elemente der übergeordneten Komponente LKrankenhaus zugeordnet. Weiterhin wird die separate Lokation LRechenzentrum nicht weiter betrachtet, da deren Dienstleistungen indirekt über entsprechende Wrapper-Agenten angeboten werden, die aber aus technischen Gründen[56] nicht direkt auf den dortigen Maschinen ihre Funktionalitäten zur Verfügung stellen können. Auch auf die separate Modellierung des Patienten in einer eigenen Plattform wird verzichtet, weil bei der Modellierung des ausgewählten Prozessausschnittes nur die Aggregation der Patientendaten betrachtet wird, jedoch nicht die selbstständige Informationsgewinnung durch den Patienten.

6.6.4.2 Bestimmung und Klassifikation von Tasks

Wie bei *Reinke* (2003, 92-93) beschrieben, sind die zu identifizierenden Tasks nach den Typen WrapperTask, UserAssistanceTask, ManagementTask, InteractionTask und ActionTask zu klassifizieren (siehe Reinke 2003, 85). Mit der Interaktion zwischen zwei Agenten wird eine Task vom Typ InteractionTask assoziiert. Die jeweilige Rolle in der Interaktion wird in der Nomenklatur durch die Bezeichnung „Send“ und „Receive“ für die jeweilige Interaktionsseite gekennzeichnet (siehe auch Bastian 2005, 90).

[56] Für die vorliegende Arbeit stand keine experimentelle Hardware zur Verfügung, auf der die Wrapper-Agenten installiert werden konnten. Da aus der Sicht der Applikation von der Lokation der Wrapper-Agenten abstrahiert wird, stellt diese Implementierungsbeschränkung keine qualitative Reduktion der Ergebnisse der vorliegenden Arbeit dar.

Die Basis für die Bestimmung und Klassifikation von Tasks bildet die in Abschnitt 6.6.2 beschriebene Domänenmodellierung, insbesondere die dabei entwickelten Prozessmodelle. Die Ergebnisse dieser Typisierung sind in Tabelle 6.6-3 zusammengefasst. Die ausführliche Beschreibung des Verarbeitungsmodells und der jeweiligen Tasks erfolgt in Abschnitt 6.8.1.

Action/Interaction	**Task**	**Tasktyp**	**Beschreibung**
Patientenakte öffnen	• SendOpenEHR • ReceiveOpenEHR	InteractionTask	Öffnen der Patientenakte einschließlich interner Agenten
Patientenakte suchen	lookUpEHR	ActionTask	Suche nach der Patientenakte
Suchanfrage abschicken	• SendQueryPatientData • ReceiveQueryPatientData	InteractionTask	Die Suchanfrage nach den bisherigen Dokumenten zum Patienten wird an das AMD gesendet
Zugriff überprüfen	checkAccess	ActionTask	Überprüfung, ob der Zugriff erlaubt ist
Ressourcen über eGK lokalisieren	• SendLocateResources • ReceiveLocateResources • SendRequestLinks • ReceiveRequestLinks	InteractionTask	Auslesen und Weiterleiten der Verweisliste über den Konnektor aus der TI
Suchagenten generieren	GenerateTaskAgent	ActionTask	Suchagenten für jeweils eine Ressource aus der Verweisliste generieren
Warten auf die Ergebnisse der Suchanfrage	• ReceiveRequestPatientData • ReceiveDeliverDocuments	InteractionTask	Warten der Task-Agenten auf Anfragen zur Abfrage der Patientendaten bzw. auf die abgefragten Inhalte
Weiterleitung der Anfragen an die Wrapper-Agenten bzw. den Konnektor	• SendQueryDocuments • SendRequestPatientData	InteractionTask	Die Anfragen werden entsprechend der Lokation an die jeweiligen Agenten weitergeleitet.
Empfangen der Anfrage der internen Ressource	• ReceiveQueryDocuments • SendDeliverDocuments	InteractionTask	Die Anfrage zur Extraktion von Daten aus dem internen Informationssystem wird empfangen bzw. mit dem Ergebnis an den Anfrager zurückgeschickt.
Weiterleitung von Anfragen an externe Ressourcen	• SendReceiveExternalRequestBehaviour • ReceiveSendExternalRequestBehaviour	InteractionTask	Weiterleitung der Anfrage zur Extraktion von Daten aus entfernten Informationssystemen

Empfangen einer externen Anfrage	ReceiveRequestExternalData	WrapperTask	Empfangen einer Anfrage aus einem externen System
Suchanfrage ausführen	executeQuery	WrapperTask	Die Anfrage wird in einen Aufruf an das Informationssystem umgewandelt.
Weiterleiten der extrahierten Dokumente	• SendAddDocuments • ReceiveAddDocuments	InteractionTask	Die extrahierten Dokumente werden an den Zielort weitergeleitet.
Patientendaten aufbereiten	ProcessDataToGUI	ActionTask	Grafische Aufbereitung der empfangenen Daten für die Anzeige an der Benutzeroberfläche
Patientendaten anzeigen	addUpdate	UserAssistanceTask	Anzeige der Daten an der Benutzeroberfläche

Tabelle 6.6-3: ***Bestimmung und Klassifikation von Tasks***
Quelle: In Anlehnung *Bastian* (2005, 90-91)

6.6.4.3 Gruppierung von Tasks und Bestimmung von Agentenklassen

Entsprechend der bei *Reinke* (2003, 94-95) erforderlichen Gruppierung von Tasks und Klassifizierung von Agenten wird dieser Entwicklungsschritt in diesem Abschnitt beschrieben. Die Ergebnisse davon sind in Tabelle A.4-1 dargestellt. Zu bemerken ist dabei, dass in der genannten Ergebnistabelle die Tasks nur auszugsweise dargestellt sind. Aus Gründen der Übersichtlichkeit wird zur detaillierten Beschreibung auf Anhang A.4 verwiesen. Dort wird die Verarbeitungsvorschrift der jeweiligen Agenten dargestellt.

Bei der Bestimmung der Agentenklassen wird, entsprechend einer geeigneten, abstrakten Musterarchitektur (siehe Abbildung 6.6-12), die an der Architektur für dialogorientierte Applikationen (Horn/Schubert 1993) ausgerichtet ist (Reinke 2003, 31), zwischen den Teilsystemen User Interfaces, Application Core, External Resources und Platform Services differenziert (Reinke 2003, 31-33):

- *User Interface*: Die Benutzerschnittstelle wird von den Agenten User Assistants (Interaktion mit dem Anwender), User Inquiring Assistants (Akquise von zur Erfüllung delegierter Aufgaben erforderlicher Daten) und Administrator Assistants (Management der Agentenplattform) gebildet.
- *Application Core*: Im Anwendungskern wird die domänenspezifische Funktionalität abgebildet. Diese wird von Management Agents sowie von Task Agents realisiert. Erstere delegieren Aufträge an Tasks Agents und koordinieren diese, letztere führen die eigentlichen Aufgaben aus.
- *External Resources*: Die Integration von externen Ressourcen wird durch Wrapper Agenten vorgenommen. Dabei können unterschiedliche Datenquellen wie z.B. Datenbanken, Altsysteme, Dateisysteme oder Netzwerkdienste differenziert werden.

- *Platform Services*: Da die Plattformdienste von der gewählten Agentenplattform realisiert werden (siehe Abschnitt 6.3.2 für eine Darstellung der JADE-Plattform), sind diese unabhängig von der in der vorliegenden Arbeit zu konstruierenden Applikation und werden deshalb nicht weiter betrachtet.

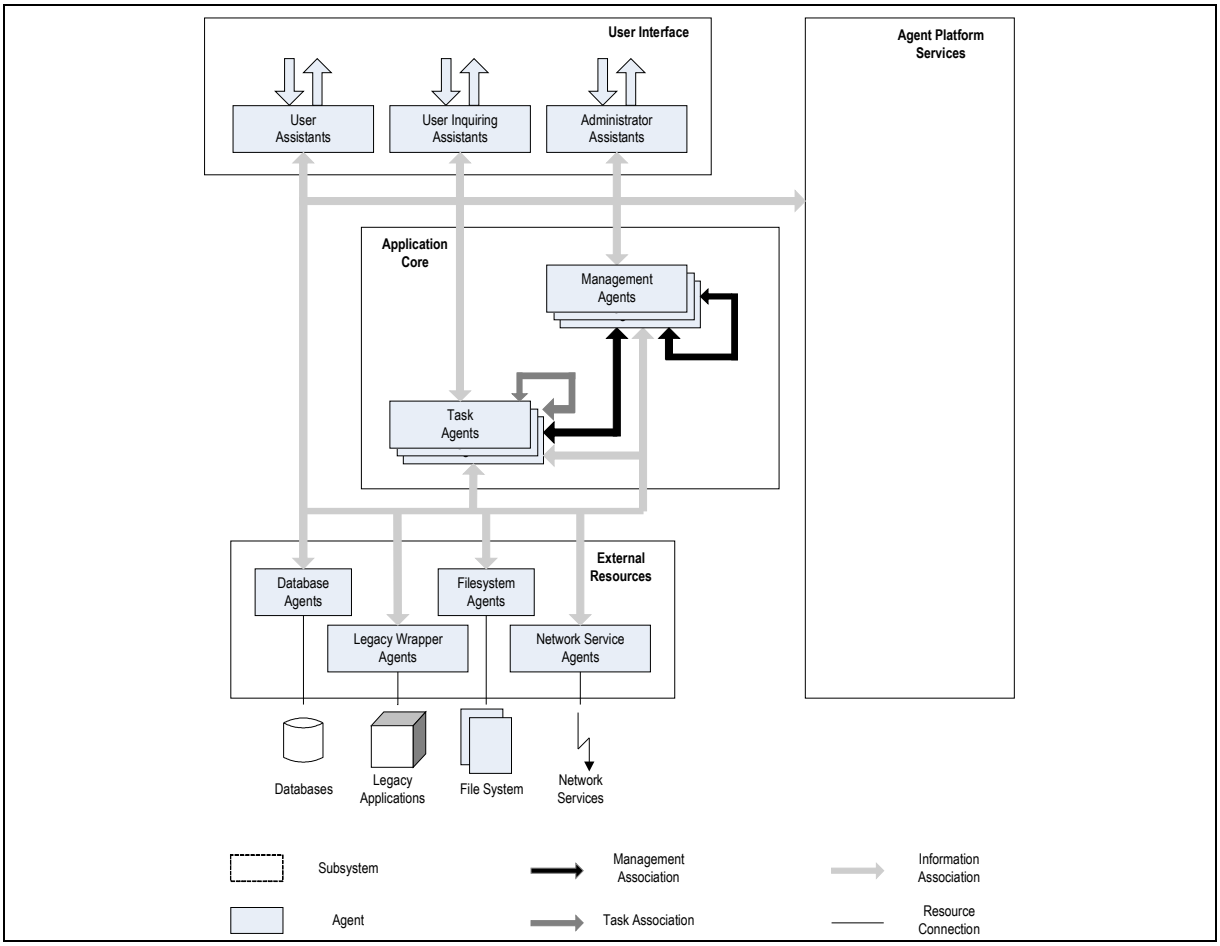

Abbildung 6.6-12: Allgemeine Musterarchitektur für Multiagentensysteme
Quelle: In Anlehnung an *Reinke* (2003, 31)

Die gewählte Musterarchitektur mit den Teilsystemen Application Core, External Resources und User Interfaces reflektiert die Eigenschaften einer Schichtenarchitektur (siehe dazu Abschnitt 3.2.3.1). Somit wird das etablierte Muster der Schichtenarchitektur auf Agentensysteme übertragen, indem letztere in Architekturen hineinentwickelt werden (Anforderung 16 aus Abschnitt 6.4.2).

Abschließend ist noch darauf hinzuweisen, dass die Verfeinerung der in Abschnitt 6.6.4.2 identifizierten Tasks, die eigentlich in Etappe 3 vorgesehen ist (Reinke 2003, 119-120), bereits in Etappe 2 vorgezogen wurde (siehe dazu auch Bastian 2005, 91). Diese verfeinerten Tasks können damit bereits in die weitere Modellierung der nachfolgenden Etappen einfließen.

6.6.4.4 Zusammenfassung weiterer Modellierungsergebnisse der Etappe 2

Nach den Ausführungen bei *Reinke* (2003, 97-100) sind für die bislang beschriebenen Tasks Interaktionsmodellierungen vorgesehen. Weil in Abschnitt 6.8.1 diese Interaktionen ausführlich beschrieben werden, wird in diesem Abschnitt auf ihre Darstellung verzichtet.

Im Folgenden werden die Modellierung von Konnektoren sowie die Zusammenfassung der bisherigen Ergebnisse in einer spezialisierten Musterarchitektur (Abbildung 6.6-13) beschrieben:

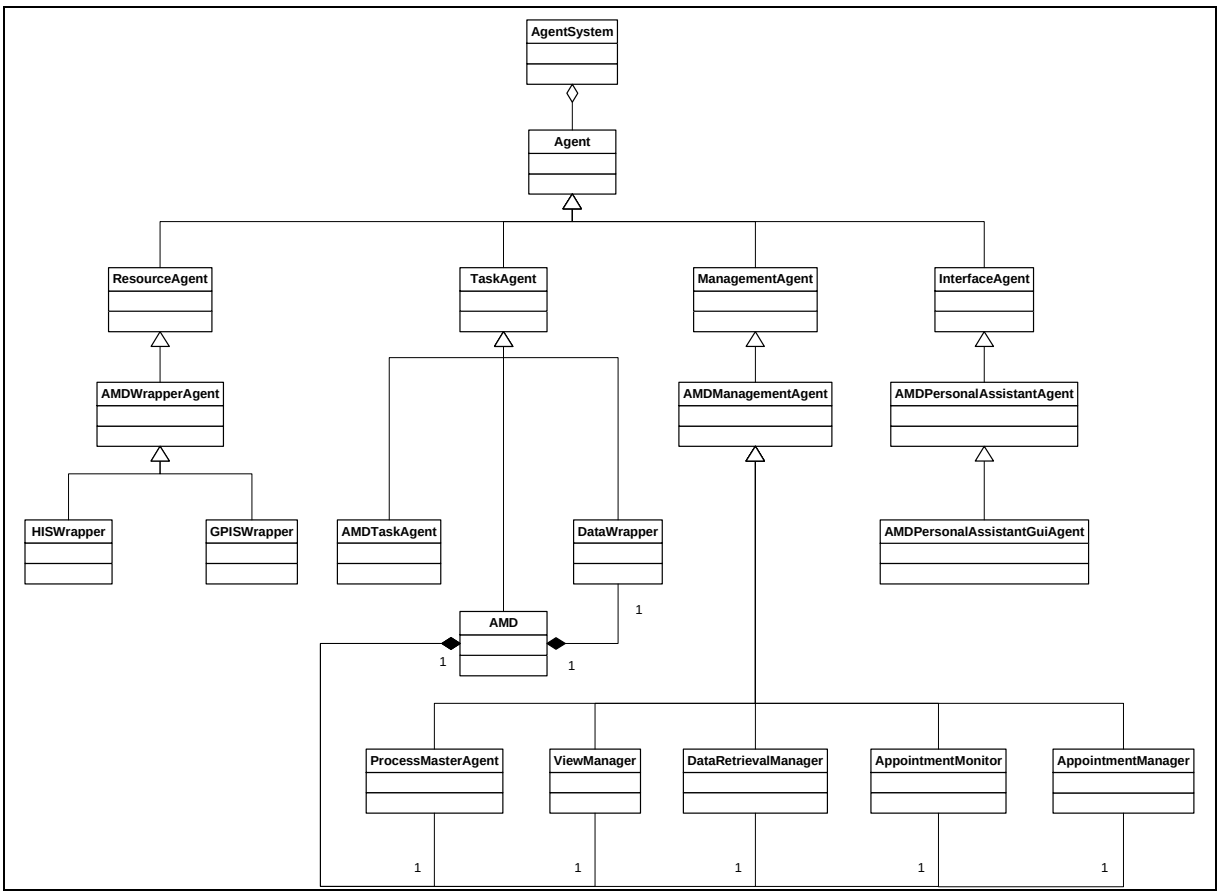

Abbildung 6.6-13: ***Modelltyp für die spezialisierte Architektur***
Quelle: In Anlehnung an *Bastian* (2005, 97)

Die explizite Berücksichtigung von Konnektoren (für ihre Bestimmung in der Etappe 2 siehe Reinke 2003, 100-102) ist in der für die vorliegende Arbeit gewählten JADE-Plattform nicht gegeben. Deshalb wird auf die gezielte Darstellung und Modellierung von Konnektoren verzichtet. Stattdessen werden die durch diese Komponenten repräsentierten Kommunikationspunkte zwischen Agenten entsprechend den Ausführungen bei *Reinke* (2003, 132-134) direkt auf Behaviours von Agenten abgebildet, wobei zwischen anfordernden oder sendenden und antwortenden Verhaltensmustern differenziert wird (siehe dazu Reinke 2003, 132).

Als Abschluss der Modellierung in Etappe 2 werden die bisherigen Ergebnisse in einem Modelltyp (Reinke 2003, 104-105) für die auf der Grundlage einer allgemeinen Musterarchitek-

tur (siehe Abschnitt 6.6.4.3) spezialisierten Architektur zusammengefasst (siehe Abbildung 6.6-13). Bei Abbildung 6.6-13 ist darauf hinzuweisen, dass die darin dargestellten Agenten ResourceAgent, TaskAgent, ManagementAgent und InterfaceAgent nur das Konzept reflektieren, aber keine direkte Abbildung in der Implementierung erfahren.

6.6.5 Modellierung in der Etappe 3

Die Etappe 3 umfasst nach der Ausführung von *Reinke* (2003, 107-142) die Konstruktion und das Testen der bisher modellierten Teilsysteme. Dabei werden u.a. die folgenden Schritte durchgeführt:

- Interaktionsmodellierung einschließlich der dabei ausgetauschten Nachrichten (Reinke 2003, 116-117)
- Verfeinerung von Tasks sowie die Entwicklung von Architekturmodellen für die jeweiligen Agenten (Reinke 2003, 121-128)
- Abbildung auf eine konkrete Agentenplattform (Reinke 2003, 128-134)
- Anbindung von Benutzerschnittstellen (Reinke 2003, 138-139) und externen Ressourcen (Reinke 2003, 139-140)

Für die Beschreibung der Interaktionsmodellierung wird auf die Ausführungen in Abschnitt 6.8.1 verwiesen. Die Verfeinerung von Tasks wurde bereits in Abschnitt 6.6.4 vorgenommen. Die Architekturmodelle für zentrale Agenten sind in Anhang A.5 dargestellt. Die Abbildung auf eine Agentenplattform sowie die Integration von externen Ressourcen werden im Folgenden beschrieben:

Entsprechend den Kernpunkten der Model Driven Architecture[57] (MDA®) folgt auch die ArBaCon-Methode der Abbildung von Plattform unabhängigen Modellen (Platform Independent Model, PIM) zu Plattform spezifischen Modellen (Platform Specific Model, PSM): Ausgangspunkt für diese Transformationsschritte ist bei der ArBaCon-Methode die Wahl einer geeigneten, abstrakten Musterarchitektur (siehe Abbildung 6.6-12, PIM), welche entsprechend den Anforderungen der Domäne in eine spezialisierte Architektur (PIM mit domänenspezifischen Eigenschaften) überführt wird. Letztere wird durch die Implementierung der Applikation auf eine bestimmte Agentenplattform (PSM) abgebildet (siehe dazu die Ausführungen bei Reinke 2003, 47-48).

Für die vorliegende Arbeit ist die Abbildung der bisher beschriebenen Modelle auf die gewählte Agentenplattform JADE (siehe dazu Abschnitt 6.3.2) relevant. Die dafür erforderliche Abbildung von Aktivitäten auf Behaviours bzw. Operationen ist in Abbildung 6.6-14 dargestellt und wird im Folgenden beschrieben:

Ausgehend von der Differenzierung, ob eine interaktive oder reguläre Aktivität vorliegt, werden InteractionTasks bzw. ManagementTasks, UserInterfaceTasks, WrapperTasks oder ActionTasks gewählt. Komplexe Management-Tasks sind für eine Verfeinerung (Delegation von Aufgaben an andere Agenten sowie ihre Koordination, insbesondere auch zur Laufzeit, Anforderung 18

[57] *Object Management Group* (2000; 2001a; 2001b; 2003), *Kleppe/Warmer/Bast* (2003).

aus Abschnitt 6.4.2) prädestiniert: Für Interaktionen ist ein Kommunikationspunkt (Port) erforderlich, welcher entsprechend den Ausführungen in Abschnitt 6.6.4.4 auf sendendes bzw. empfangendes Verhalten abgebildet wird.

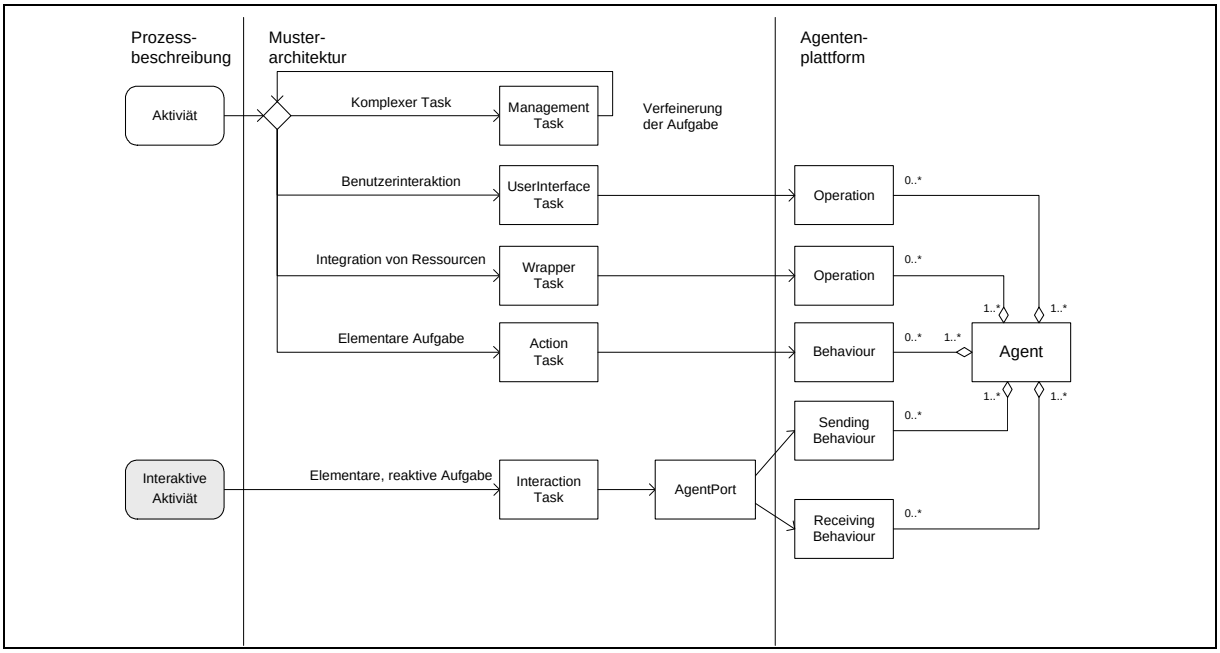

Abbildung 6.6-14: ***Transformationsmodell***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Für die Benutzerinteraktion bzw. die Integration von Ressourcen erweist sich die Wahl von Operationen anstatt Behaviours als vorteilhaft. Diese Entscheidung ist aus den folgenden Gründen gerechtfertigt:

Die Komplexität, welche sich durch die Synchronisation der unterschiedlichen Tasks ergeben würde, wenn für die Implementierung der Benutzerinteraktion Behaviours eingesetzt würden, kann erheblich reduziert werden. Außerdem kann insbesondere die Benutzerschnittstelle durch bereits vorhandene Programmiermodelle umgesetzt werden: So stellt die in der Implementierung eingesetzte Sprache Java (Anforderung 12 aus Abschnitt 6.4.2) Event- und Listener-Funktionalität in einer umfangreichen Bibliothek zur Verfügung. Würde die Benutzerinteraktion, so wie es von dem Programmiermodell bei JADE vorgesehen ist (Bellifemine et al. 2005, 17-20), über das eigene Verarbeitungsmodell von JADE umgesetzt, wären umfangreiche Fallunterscheidungen beim Abarbeiten der aufgetretenen Ereignisse erforderlich, die die Komplexität und damit die Fehleranfälligkeit der Implementierung erhöhen würden. Außerdem müssten damit neben dem bereits vorhandenen Java-Verarbeitungsmodell für die Benutzerinteraktion eigene Implementierungen für die Benutzerinteraktionen umgesetzt werden. Dadurch würde der Implementierungsaufwand weiter zunehmen. In dem JADE-Modell für die Bearbeitung von Ereignissen wird zudem zwischen dem Java-Ereignismodell und der Ereignisverarbeitung eine weitere Schicht hinzugefügt, die wiederum den Implementierungsaufwand und damit auch die Fehleranfälligkeit erhöht: Die Java-Ereignisse müssten in JADE-GUI-Ereignisse transformiert und in einem anschließenden Schritt über Fallunterscheidungen verarbeitet werden. Diese Fallunterscheidungen sind in einer Benutzeroberfläche mit zahlrei-

chen Interaktionsmöglichkeiten umfangreich. Aus den genannten Gründen wird in der in der vorliegenden Arbeit beschriebenen Implementierung auf das Verarbeitungsmodell von JADE für die Benutzerinteraktion verzichtet. Stattdessen werden die vorhandenen Java-Bibliotheken verwendet.

Für die Implementierung der Benutzerschnittstelle wird in der Architekturschicht User Interface (siehe dazu Abschnitt 6.6.4.3) das etablierte Muster MVC (siehe dazu Abschnitt 3.2.3.1) eingesetzt. Dazu erfolgt die Aufteilung der Funktionalitäten in Klassen zur Modelldarstellung, Visualisierung der Benutzeroberfläche und Steuerung der Benutzerinteraktion. Zur Modularisierung der Komponenten für die Realisierung der Benutzerschnittstelle trägt weiterhin bei, dass die zu visualisierenden Texte über eine separate Komponente aus einem Repository mit den Begrifflichkeiten zur Verfügung gestellt werden. Eine Anpassung der Anwendung an unterschiedliche Sprachen ist damit über eine zentrale Komponente realisiert und gestaltet sich deshalb einfach.

Altsysteme und andere, nicht agentenbasierte Komponenten werden mit speziellen Wrapper-Agenten (Jennings 2001, 40) in das Agentensystem integriert (siehe dazu auch Abschnitt 6.5.2). Die dafür relevanten Aspekte werden im Folgenden beschrieben:

Entsprechend dem Wrapper-Konzept werden in dem in dieser Arbeit beschriebenen Prototyp exemplarisch Informationssysteme in das Agentensystem integriert. Als Vertreter von Krankenhausinformationssystemen wird das System SAP IS-H bzw. i.s.h.med über die BAPI-JCo-Schnittstelle in das Agentensystem integriert (Anforderung 8 aus Abschnitt 6.4.1). Aus diesem System werden ausgewählte Stammdaten des Patienten extrahiert und schließlich an der Benutzeroberfläche visualisiert. Die übrigen Stammdaten werden im PersonalAssistantGuiAgent aus dem ersten empfangenen medizinischen Dokument extrahiert und angezeigt.

Die Extraktion eines EKG-Befundes aus dem KIS wird über eine BAPI-Schnittstelle simuliert. Ein Überblick über die in der Anwendung zur Verfügung gestellten BAPIs und ihrer jeweiligen Funktionalität ist in Tabelle 6.6-4 zusammengefasst. Die eigentlichen Daten für die medizinischen Dokumente werden nicht aus dem SAP-System extrahiert, sondern aus einer im Dateisystem abgelegten Datei im HL7 CDA-Format (Anforderung 4 aus Abschnitt 6.4.1) gelesen. In das CDA-Dokument sind Codierungen nach LOINC eingebettet (Anforderung 6 aus Abschnitt 6.4.1). Weiterhin werden Daten im Bildformat in dieses Dokument integriert (siehe dazu Abschnitt 6.8.2).

Aus einem PVS werden weitere Dokumente extrahiert: Die Datenaggregation wird jedoch simuliert, um die Komplexität der Implementierung zu reduzieren. Das Ergebnis der Anfrage an das PVS wird aus lokal abgelegten Dateien im HL7 CDA-Format ausgelesen. Ein radiologischer Befund, d.h. Daten im Bildformat, ist in das CDA-Dokument (siehe dazu Abschnitt 6.8.2) eingebettet. Weiterhin wird von dem simulierten PVS ein Arztbrief zur Verfügung gestellt, wobei auch hier LOINC-Codierungen enthalten sind.

BAPI	Methodenimplementierung	Funktionalität
BAPI_PATIENT_SEARCH	searchPatientBAPI	Übergeben wird eine externe Patientenidentifikation, die der auf der eGK gespeicherten Identifikation entspricht. Aus dem SAP-System wird für den gewählten Patienten die interne Patientenidentifikation zurückgeliefert.
BAPI_PATIENT_GETDETAIL	getPatientDetailBAPI	Übergeben wird die interne Patientenidentifikation des SAP-Systems. Zurückgeliefert werden die Stammdaten des Patienten.
BAPI_PATIENT_GETCASELIST	getPatientCaseListBAPI	Übergeben wird die interne Patientenidentifikation des SAP-Systems. Zurückgeliefert wird die Fallliste für diesen Patienten mit allen bisherigen Fällen.
BAPI_CASE_DIAGNOSIS_GETLIST	getCaseDiagnosisListBAPI	Übergeben wird die interne Fallnummer des SAP-Systems. Zurückgeliefert wird eine Liste von Diagnosen zu diesem Fall.
BAPI_CASEFINDINGS_GETLIST (simuliert als BAPI)	getCaseFindingsListBAPI	Übergeben werden die Fallnummer und die interne Patientenidentifikation des SAP-Systems. Die Implementierung der BAPI ist nur simuliert. Es werden die Befundlisten nicht aus dem SAP-System ausgelesen, weil eine entsprechende BAPI in der für diese Arbeit vorliegenden Version des KISs nicht implementiert ist. Vielmehr werden die Befunde aus Dateien des lokalen Dateisystems ausgelesen. Wegen der implementierten Schnittstelle ist eine Anbindung an das KIS bei Verfügbarkeit einer umfassenderen BAPI-Schnittstelle mit geringem Aufwand erforderlich.

Tabelle 6.6-4: ***Übersicht über relevante BAPIs***
Quelle: Eigene Darstellung

Die aus den Informationssystemen extrahierten medizinischen Dokumente werden im Content-Slot einer ACL-Nachricht (Anforderung 7 aus Abschnitt 6.4.1) zu ihrer weiteren Verarbeitung versendet. Zur Visualisierung an der Benutzeroberfläche wird entsprechend einer geeigneten Stylesheet-Definition aus der HL7 CDA-Datei im XML-Format eine HTML-Datei generiert. Das Ergebnis dieser Transformation kann ohne weitere Verarbeitung in die Java-Benutzeroberfläche eingefügt werden. Dabei können die Eigenschaften der Java Swing-Fähigkeiten vorteilhaft ausgenutzt werden.

Um die Implementierungskomplexität zu reduzieren, wird die weiter oben beschriebene Extraktion von Daten aus den gewählten Informationssystemen nicht über Behaviours realisiert, sondern über Operationen abgebildet.

Für die in Etappe 3 vorgesehenen Tests wird auf die Ausführungen in Abschnitt 7.4 verwiesen, in dem die durchgeführten Tests beschrieben werden.

6.6.6 Modellierung in der Etappe 4

Im Zentrum der Etappe 4 steht bei *Reinke* (2003, 143-162) das Deployment der implementierten Applikation. Dabei werden die in den vorhergehenden Etappen konstruierten Teilsysteme auf Plattformen bzw. Rechnerknoten verteilt. Dazu sind nach *Zendler* (1997; Reinke 2003, 144) folgende Schritte abzuarbeiten:

- Auswahl von verteilbaren Elementen und die Zuweisung von realisierten Komponenten zu diesen Elementen
- Allokation der verteilbaren Elemente auf physikalische Knoten und Laufzeitumgebungen (Deployment)

Für die Verteilung dieser Elemente können folgende Kriterien herangezogen werden[58]:

- *Restriktion*: Eine ausreichende Kapazität auf dem Zielrechner ist Voraussetzung für die Zuordnung von verteilbaren Elementen oder Agentenplattformen.
- *Präferenz*: Verteilbare Elemente sind bestimmten Agentenplattformen und physikalischen Knoten zuzuordnen (z.B. zur Kapselung von Ressourcen aus dem nicht agentenbasierten Bereich).
- *Exklusivität*: Teilmengen von verteilbaren Elementen mit jeweils gegenseitig einschränkenden Eigenschaften dürfen nicht auf gleiche Rechner bzw. Agentenplattformen verteilt werden (z.B. zur Sicherstellung der Performanz).
- *Redundanz*: Werden verteilbare Elemente häufig oder an verschiedenen Lokationen verwendet, sind sie aus Performanzgründen verschiedenen physikalischen Knoten oder Agentenplattformen zuzuordnen.
- *Affinität*: Besteht zwischen verteilbaren Elemente eine enge Beziehung, sind diese auf dem gleichen physikalischen Knoten oder der gleichen Agentenplattform zu allokieren.

Entsprechend diesen Ausführungen wird im Folgenden die Allokation für die vorliegende Applikation beschrieben (siehe dazu auch Abbildung 6.6-15): Auf der Plattform Telematics sind nicht die Agenten zur Implementierung der Konnektoren allokiert, sondern es sind dort lediglich ihre Dienste registriert. Damit stehen diese Dienste auch anderen Agenten zur Verfügung, um die Interoperabilität zu gewährleisten. Außerdem wird damit die Eigenschaft der TI berücksichtigt, nach der die Konnektoren die Grenze der Infrastruktur darstellen und einen Teil davon bilden. Dennoch sind die Konnektoren physikalisch in den Bereich der Primärsysteme einzuordnen. In der prototypischen Implementierung ist eine einzige Plattform für die Abbildung der TI ausreichend, für den Einsatz in der Praxis ist aus Performanzgründen eine adäquate Skalierung vorzusehen.

[58] *Ma/Lee/Tsuchiya* (1982), *Yan/Lundstrom* (1998), für eine Zusammenfassung siehe *Zendler* (1997), für eine Übertragung der Kriterien auf MASe siehe *Reinke* (2003, 159).

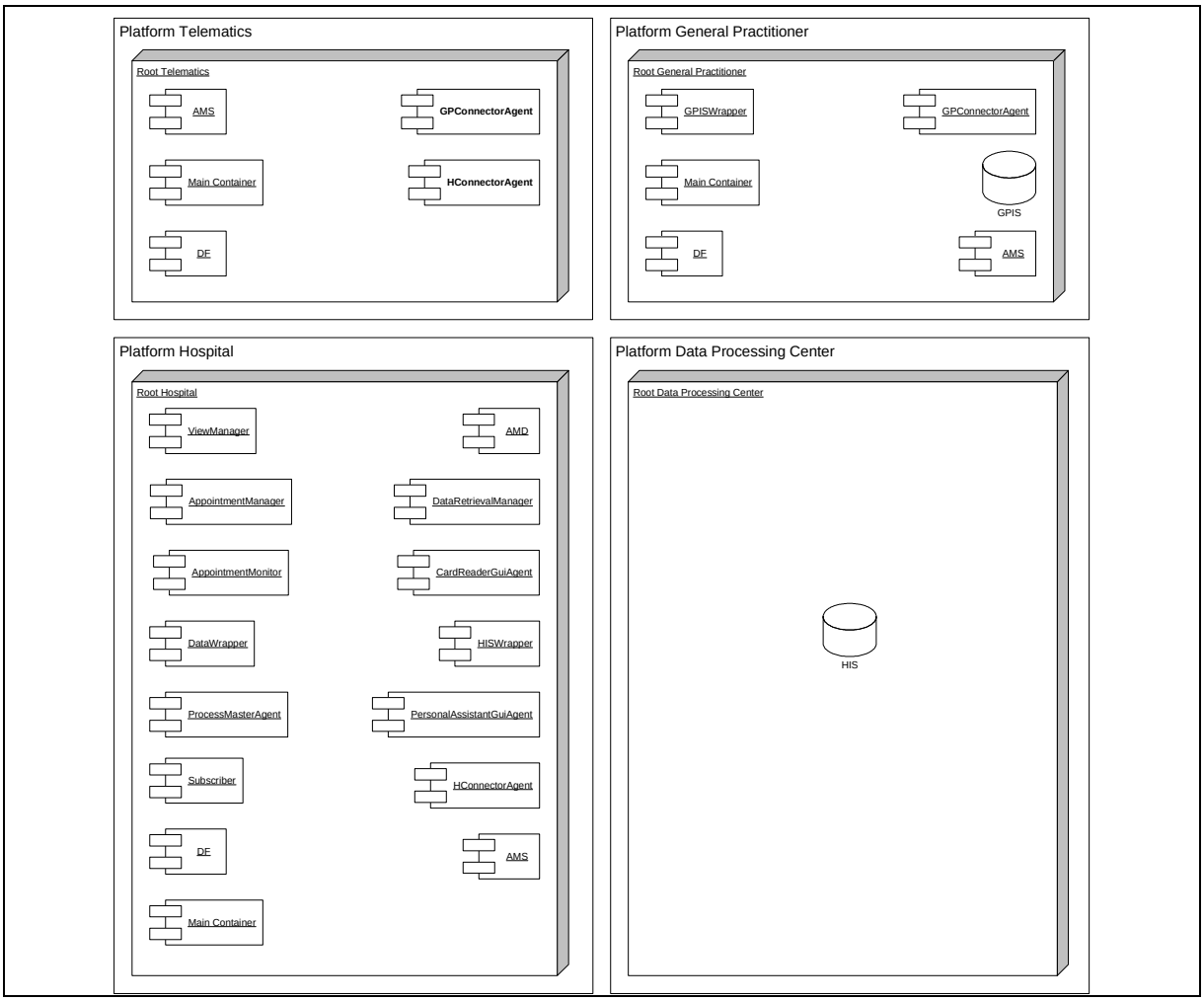

Abbildung 6.6-15: Deployment der Anwendung
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Wrapper-Agenten, die externe Ressourcen kapseln, sind nach dem Prinzip der Präferenz jeweils denjenigen Plattformen zuzuordnen, auf denen die zu kapselnden Systeme installiert sind. Aus technischen Gründen (siehe dazu Abschnitt 6.6.4.1) konnten diese Agenten nicht direkt auf der KIS-Plattform installiert werden. Für eine prototypische Implementierung ist es aber ausreichend, den Wrapper-Agenten für das KIS ebenfalls der Hospital-Plattform zuzuordnen. Für ein korrektes Deployment auf der Plattform Data Processing Center wäre keine zusätzliche Funktionalität erforderlich, sondern lediglich eine Änderung im Deployment. Somit kann eine entsprechende Anpassung vereinfacht durchgeführt werden. Das AMD einschließlich seiner zugehörigen Agenten (Prinzip Affinität) wird je nach Anwendungsszenario auf die Plattformen Hospital oder General Practitioner allokiert, weil diese Dokumente häufig genutzt werden (Prinzip Redundanz). In dem in dem Fallbeispiel fokussierten Szenario wird die AMD-Funktionalität im Referenzklinikum benötigt und damit der Plattform Hospital zugewiesen. Die Benutzerschnittstellen für das AMD sind derjenigen Plattform zugeordnet, auf die die jeweiligen Nutzer Zugriff besitzen (Prinzip Präferenz). Damit werden für das gewählte Szenario CardReaderGuiAgent (Kartenleser) und PersonalAssistantGuiAgent (Darstellung der virtuellen Patientenakte) jeweils mit ihrer Benutzeroberfläche auf der Plattform Hospital allokiert.

Nach den Prinzipien der Redundanz und Affinität werden die Konnektoren (HConnectorAgent und GPConnectorAgent) jeweils der Plattform Hospital bzw. General Practitioner zugeordnet. Die Affinität ist gegeben durch die Kommunikation mit dem AMD auf der jeweiligen Plattform. Mit der gleichen Begründung wie für die Zuordnung des HISWrappers zu der Plattform Hospital (Prinzip Präferenz) wird auch der Wrapper für das PVS (GPISWrapper) der Plattform General Practitioner zugeordnet.

Das in Abschnitt 6.6.3.3 beschriebene Lokationsmodell wird nicht in allen Einzelheiten in dem in diesem Abschnitt dargestellten Deployment berücksichtigt. Die Begründungen dazu sind in Abschnitt 6.6.4.1 ausgeführt.

In Abschnitt 6.6 wurde die Anwendung der Methode ArBaCon für den Anwendungsfall der Aggregation von Patientendaten während der OP-Vorbereitung im Rahmen der Behandlung des kolorektalen Karzinoms im gewählten Referenzklinikum beschrieben. Damit kann die agentenbasierte Implementierung der Informationsaggregation aus verteilten und heterogenen Informationssystemen vorgenommen werden.

Im folgenden Abschnitt 6.7 wird das Design für die übrigen Anwendungen beschrieben, die im Anforderungskatalog in Abschnitt 6.3.6 genannt sind.

6.7 Beschreibung des Designs

Nach der Modellierung im vorausgehenden Abschnitt 6.6 mit dem Schwerpunkt der Informationsaggregation in der virtuellen Patientenakte wird in den folgenden Ausführungen das Design der Applikationen mobile Anwendung, Scheduling, Patientenportal sowie Prozessunterstützung beschrieben.

6.7.1 Desktop-Applikation zur Informationsaggregation

Die Desktop-Anwendung mit ihrem zentralen Element des AMD (siehe Abschnitt 6.5.1) kann als der tragende Teil der Implementierung betrachtet werden, in den weitere Applikationen modular integriert werden können. Deshalb wurden für diese Anwendung die einzelnen Entwurfsschritte ausführlich beschrieben. Diese folgen der bei *Reinke* (2003) beschriebenen Methode zur architekturbasierten Konstruktion von Multiagentensystemen (siehe dazu auch Abschnitt 1.7). Diese detaillierten Entwurfsschritte sind bereits in den Abschnitten 6.6.2 mit 6.6.6 dargestellt. Deshalb wird in diesem Abschnitt nicht erneut das Design für diese Applikation aufgegriffen.

6.7.2 Einbettung in die Telematikinfrastruktur

Zur Berücksichtigung von Eigenschaften der TI für die eGK werden für die prototypische Umsetzung insbesondere diejenigen Elemente ausgewählt, die für die Funktionalität der virtuellen Patientenakte erforderlich sind. Dabei ist das Ziel weniger die korrekte Berücksichtigung aller Details der vorliegenden Spezifikationen der TI und ihrer Komponenten, sondern vielmehr die Beleuchtung von Eigenschaften, welche zur Realisierung eines Mehrwertdienstes für die virtuelle Patientenakte beitragen können. Der Überblick über die Architektur mit den für die genannte Funktionalität erforderlichen Eigenschaften ist in Abbildung 6.7-1 dargestellt. Dabei wird insbesondere die Kapselung von Elementen durch Wrapper-Agenten (Jennings 2001, 40) deutlich.

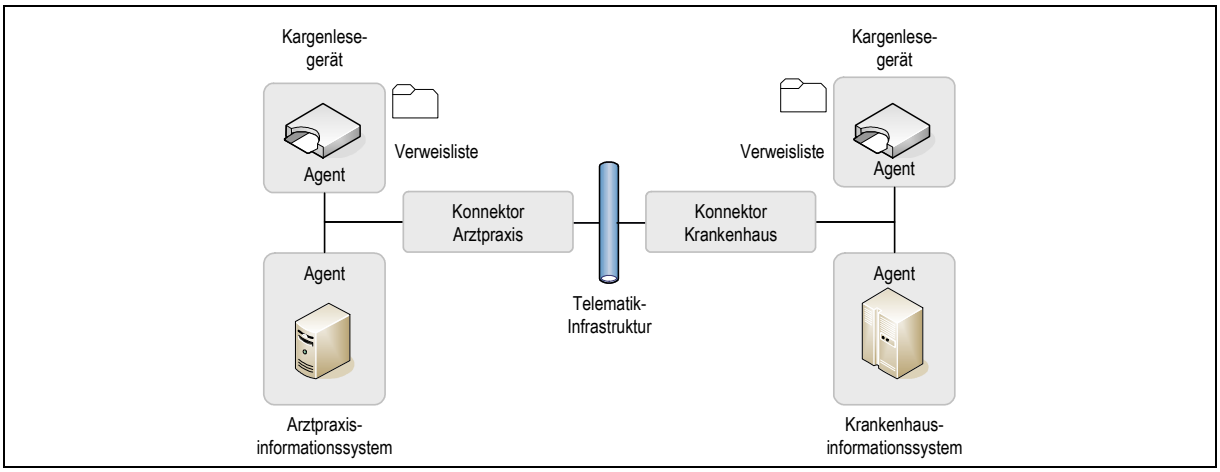

Abbildung 6.7-1: ***Berücksichtigung von Eigenschaften der TI***
Quelle: Eigene Darstellung

In dem für die Implementierung gewählten Szenario der OP-Vorbereitung werden Daten vom Standort des KISs aggregiert. Dazu wird über den Konnektor des Krankenhauses zunächst die Verweisliste aus der im Kartenlesegerät (Anforderung 2 aus Abschnitt 6.4.1) eingelegten eGK ausgelesen. Dieser Zugriff ist entsprechend den Vorgaben aus SGB V § 291a Abs. 6, (Bundesministerium der Justiz 1988) geeignet zu protokollieren (Anforderung 3 aus Abschnitt 6.4.1). Es folgt daraufhin einerseits die direkte Abfrage von Daten aus dem lokalen KIS. Die Anfrage an das entfernte Arztpraxisverwaltungssystem wird durch die TI bzw. die entsprechenden Konnektoren (Anforderung 1 aus Abschnitt 6.4.1) im Krankenhaus bzw. in der Arztpraxis mediiert. Die aus dem PVS extrahierten Daten werden in analoger Weise wieder über die beiden Konnektoren an die agentenbasierte Anwendung in der Umgebung Hospital versandt. Zu berücksichtigen ist, dass das Auslesen der Verweisliste zur Zusammenstellung der virtuellen Patientenakte in der vorliegenden Arbeit so implementiert wird, dass eine eGK sowie ein HBA und die jeweilige PIN-Autorisierung durch den jeweiligen Besitzer erforderlich sind. Obwohl bei *Bernnat* (2006, 247) wegen der fehlenden Spezifikation der Funktionalität für die elektronische Patientenakte auf der Basis der eGK angenommen wird, dass der Zugriff auf die elektronische Patientenakte auch ohne eGK über die Bereitstellung von Präsenztoken mit einer zeitlichen Beschränkung der Gültigkeit sowie der Assoziation mit den zugriffsberechtigten Akteuren erfolgt, wird in der vorliegenden Arbeit zur Reduktion der Implementierungskomplexität vorausgesetzt, dass bei jedem Zugriff auf die virtuelle Patientenakte die eGK eingelegt ist.

Weil die Kommunikation zwischen dem Konnektor und der TI für den Anwender transparent erfolgt (Broy 2005, 25), wird die Kommunikation zwischen den Konnektoren nicht durch eine weitere Komponente, die die TI simuliert, mediiert. Denn damit würde lediglich ein zusätzlicher Aufwand entstehen, der aber keinen Mehrwert hinsichtlich der gewünschten Erkenntnisse hervorbringen würde.

Weil die über die TI der eGK versendeten Daten sensibel sind, werden über diese Kommunikationsinfrastruktur geeignete Sicherheitsmechanismen zur Verfügung gestellt. Dementsprechend sind in der Implementierung im Rahmen der vorliegenden Arbeit exemplarisch ausgewählte Mechanismen berücksichtigt. Dazu werden die Funktionalitäten von JADE-S (Anfor-

derung 20 aus Abschnitt 6.4.2) eingesetzt. Entsprechend der Bestimmung von Tasks und ihrer Funktionalitäten in den Abschnitten 6.6.4.2 und 6.6.4.3 sowie den Ausführungen in diesem Abschnitt werden Kommunikationsflüsse zwischen Agenten verschlüsselt bzw. signiert, um Vertraulichkeit sowie Integrität zu gewährleisten. Die Mechanismen zur Verschlüsselung und Signatur bei den involvierten Agenten sind in Abbildung 6.7-2 dargestellt. Dabei wird die Kommunikation zwischen dem Kartenleser (CardReaderGuiAgent) und dem Konnektor (HConnectorAgent) sowie zwischen den Konnektoren (HConnectorAgent und GPConnectorAgent) verschlüsselt und signiert.

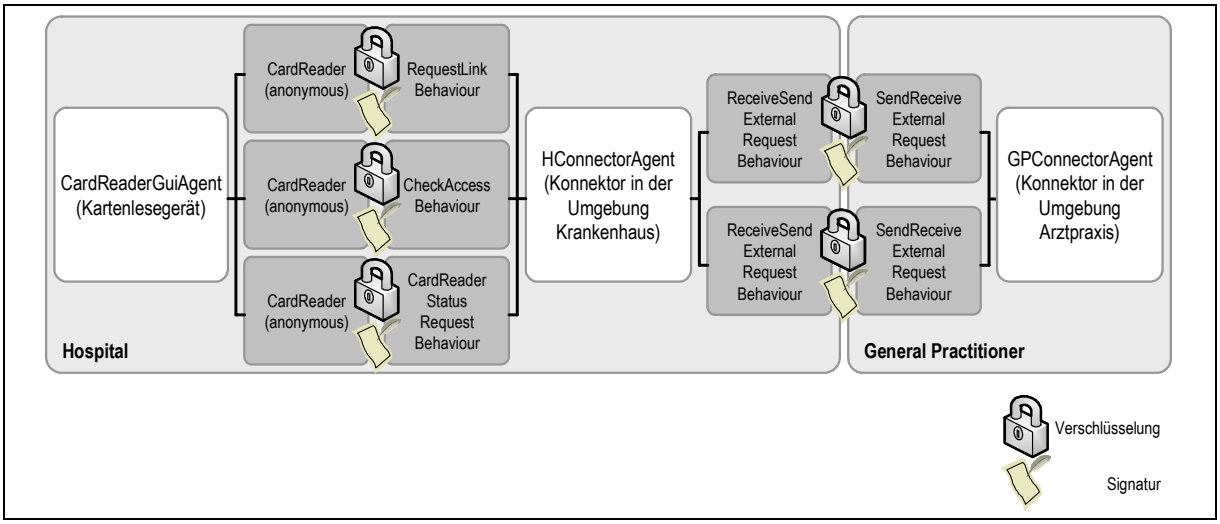

Abbildung 6.7-2: ***Design Verschlüsselung und Signatur***
Quelle: Eigene Darstellung

6.7.3 Mobile Anwendung

6.7.3.1 Überblick

Vor der detaillierten Darstellung des Designs der mobilen Applikation wird in diesem Abschnitt ein kurzer Überblick über die implementierte Funktionalität gegeben. Diese entspricht im Wesentlichen der Informationsaggregation, wie sie in der Desktop-Applikation vorgenommen werden kann.

Für eine korrekte Funktionsweise wird vorausgesetzt, dass sowohl die eGK als auch der HBA in der Desktop-Anwendung eingelegt sind. Weiterhin erfolgt, analog zur Desktop-Applikation eine Authentifizierung des Benutzers an der mobilen Anwendung. Dazu wird vorausgesetzt, dass der Anwender und die durch den eingelegten HBA bestimmte Person übereinstimmen.

Nach dieser Anmeldung kann vom Benutzer die Patientenidentifikation der eingelegten eGK übernommen werden. Diese Datenübernahme erfolgt über den Aufruf von geeigneten Funktionalitäten der Desktop-Anwendung. Es kann nun die Aggregation der virtuellen Patientenakte vorgenommen werden. Auch dabei werden die Funktionalitäten der Desktop-Anwendung verwendet. Nach der Aggregation der Daten werden diese an der Benutzeroberfläche einschließlich einer geeigneten Navigationsfunktionalität visualisiert.

Die mobile Anwendung ist weiterhin so gestaltet, dass sie zur Aggregation der virtuellen Patientenakte sowohl die beschriebene Pull- als auch eine Push-Funktionalität unterstützt. Erstere setzt die Zusammenstellung der Patientenakte über eine Benutzerinteraktion um, wie weiter oben beschrieben. Der Informationsbedarf des Anwenders wird damit gedeckt, sobald dies vom Benutzer initiiert wird. Im Gegensatz dazu ist die Push-Funktionalität so umgesetzt, dass je nach Zustand im aktuellen Behandlungsprozess für einen Patienten die Informationen automatisiert abgerufen und dann bereits zur Verfügung gestellt werden können, wenn sie benötigt werden. Daten können somit auf das mobile Endgerät gesendet werden, um die Ortsunabhängigkeit des medizinischen und pflegerischen Personals bei der Leistungserbringung zu unterstützen.

6.7.3.2 Architektur

Zur Portierung der stationären Anwendung auf ein mobiles Endgerät vom Typ XDA II (Anforderung 13 aus Abschnitt 6.4.2) und der daraus resultierenden Gestaltung der Architektur sind mehrere Architekturentscheidungen zu treffen, um die besonderen Anforderungen, die durch die beschränkten Ressourcen gegeben sind, zu berücksichtigen. Diese werden im Folgenden beschrieben und fassen die Ausführungen bei *Hillebrand* (2006, 68-73) und *Schweiger et al.* (2006a, 39-41) zusammen:

Da JADE-LEAP vollständig in Java entwickelt wurde, wird eine Java-Laufzeitumgebung auf dem mobilen Gerät vorausgesetzt. Das gewählte mobile Endgerät besitzt per se beschränkte Ressourcen. Deshalb bildet die Basis für die Java-Laufzeitumgebung auf dem mobilen Endgerät die Java MicroEdition (J2ME). Diese kann mit unterschiedlichen Konfigurationen und Profilen zusammengestellt (siehe dazu auch die Ausführungen in Abschnitt 6.3.1) und so auf Geräte mit unterschiedlichem Leistungsspektrum modular angepasst werden. Da Applikationen auf mobilen Endgeräten meist heterogen sind (siehe dazu die Ausführungen in Abschnitt 6.3.3.1), kann auf Basis der Java-Laufzeitumgebung eine entsprechende Abstraktionsschicht eingeführt werden, die den Ablauf der mobilen Anwendung auf unterschiedlichen mobilen Endgeräten ermöglicht. Im Folgenden wird die Auswahl einer geeigneten Konfigurations-Profil-Kombination beschrieben:

In dem auf dem PDA als Ausführungsumgebung installierten MIDlet-Manager[59] lässt sich bei der aktuell vorliegenden Version des ausgewählten mobilen Endgerätes kein Agent in einem JADE-LEAP-FrontEnd starten oder ausführen. Deshalb kann diese Umgebung nicht die Basis für die Implementierung darstellen.

Weil auch keine Portierung einer J2ME-Laufzeitumgebung für das Betriebssystem Windows Mobile in der Version 2003, wie es auf dem ausgewählten mobilen Endgerät vorinstalliert ist, von Sun Microsystems verfügbar ist, wird alternativ die Umgebung IBM® WebSphere® Everyplace® Micro Environment[60] eingesetzt. Die MIDP-Variante (J2ME/CLCD/MIDP) auf der Basis dieser Plattform weist jedoch bei der Integration der für das MAS relevanten

[59] Ein MIDlet-Manager gewährleistet die Ausführung und Verwaltung von MIDlets, wobei mit einem MIDlet ein Programm bezeichnet wird, das für eine J2ME/CLCD/MIDP-Umgebung implementiert wird (Hillebrand 2006, 69).

[60] Siehe dazu http://www-306.ibm.com/software/wireless/wctme, zugegriffen am 02.10.2006.

Ontologie technische Grenzen auf: Weil unter MIDP die für die Ontologie erforderliche Klasse Introspector nicht existiert, ist alternativ die Klasse MicroIntrospector zu verwenden, die die Implementierung der externalise- und internalise-Methoden in den Ontologie-Klassen voraussetzt (Hillebrand 2006, 69). Entsprechende Anpassungen in den Ontologieklassen erlaubten jedoch keine korrekte Auflösung der Vererbungshierarchien in der Ontologie (Hillebrand 2006, 69). Aus den beschriebenen Gründen stellt die J2ME/CLCD/MIDP-Plattform keine geeignete Basis für die Portierung der Applikation zur Informationsaggregation auf das gewählte mobile Endgerät dar.

Durch die Wahl einer J2ME/CDC/Personal Profile-Plattform (siehe dazu die Ausführungen in Abschnitt 6.3.1) werden die beschriebenen Defizite hinsichtlich Ontologien beseitigt. Aus diesem Grund und weil die Kombination J2ME/CDC/Personal Profile umfangreichere Grafikbibliotheken, d.h. die vollständige Funktionalität des AWTs, zur Verfügung stellt, wird für die Portierung diese Architektur gewählt. Die gegenüber der CLCD-Konfiguration höheren Hardware-Anforderungen der CDC-Konfiguration können von der vorliegenden Version des gewählten mobilen Endgerätes erfüllt werden, ohne Nachteile aus Performanzgründen in der Benutzerbedienung zu verursachen (Hillebrand 2006, 70). Entsprechende Tests bei *Hillebrand* (2006, 70) bestätigen dies.

Weil nur noch der Split-Ausführungsmodus (siehe Abschnitt 6.3.3) von JADE-LEAP bei zukünftigen Weiterentwicklungen berücksichtigt wird (Caire 2006, 6), wird dieser Modus für die vorliegende Implementierung gewählt. Wegen der Wahl der Laufzeitumgebung J2ME/CDC/Personal Profile ist auf dem mobilen Endgerät die entsprechende Version der JADE-LEAP-Plattform einzusetzen. Für das FrontEnd ist dies die PJava-Version, welche die frühere Version des Personal Profile darstellt (Hillebrand 2006, 71). Für das BackEnd wird entsprechend der dort aufgesetzten J2SE-Umgegung die Version JADE-LEAP-J2SE ausgewählt (Hillebrand 2006, 71).

Weil JADE- und JADE-LEAP-Container nicht auf der gleichen Agenten-Plattform ausgeführt werden können (Caire 2006, 6), stehen zwei Architekturvarianten zur Auswahl (Hillebrand 2006, 70-71): In Variante 1 wird der JADE-Container der Desktop-Anwendung in einen JADE-LEAP-Container transformiert. Damit würde es möglich, Agenten der JADE-LEAP- und der Desktop-Applikation in der gleichen Plattform ablaufen zu lassen. Mit dieser Entscheidung kann aber keine konzeptuelle Trennung zwischen der mobilen und Desktop-Anwendung erfolgen. Damit würde die Modularisierung der Anwendung kompromittiert werden. Deshalb wird in der Implementierung die Alternative 2 präferiert, bei der eine weitere JADE-LEAP-Plattform auf dem Desktop-Rechner hinzugefügt wird.

Die medizinischen Dokumente, die auf dem mobilen Endgerät visualisiert werden sollen, werden über ACL-Nachrichten an den PDA gesendet. Diese Dokumente folgen der HL7-CDA-Spezifikation und sind im Content-Slot der ACL-Nachrichten abzulegen (Hillebrand 2006, 71). Für die Visualisierung der HL7-CDA-Dateien in der Desktop-Anwendung genügt die Umformatierung über eine entsprechende XSLT-Transformation in ein HTML-Dokument und eine direkte Anzeige dieses HTML-Dokuments über die Swing-Funktionalität der Java-Laufzeitumgebung. Weil im Gegensatz zu einer J2SE-Umgebung in der J2ME-Plattform jedoch nur AWT-Elemente implementiert werden können, ist für die Weiterverarbeitung der

empfangenen CDA-Dokumente (Anforderung 14 aus Abschnitt 6.4.2) in der mobilen Applikation ein geeigneter Parser zu konstruieren. Weil die CDA-Dokumente semistrukturierte XML-Dokumente (Boeker/Jentzsch/Klar 2004, 1531) sind, können diese mit einem Parser auf der Basis von bestehenden Bibliotheken weiterverarbeitet werden. Die Auswahl für einen solchen Parser wird im Folgenden beschrieben:

Für J2ME-Umgebungen werden prinzipiell die folgenden Parser-Typen unterschieden (Hillebrand 2006, 42):

- *Model Parser*[61]: Das gesamte XML-Dokument wird in einem Durchlauf vollständig eingelesen und eine entsprechende temporäre Baumstruktur aufgebaut. Der Speicherbedarf bei umfangreichen Dokumenten ist entsprechend hoch, da das gesamte Dokument im Speicher vorgehalten wird.
- *Push Parser*[62]: Das Dokument wird sequenziell eingelesen. Wird ein zuvor spezifiziertes Element gefunden, wird der entsprechende Listener benachrichtigt. Auf die Elemente eines Dokuments kann also nur bei ihrem unmittelbaren Auftreten zugegriffen werden, ein wiederholter Zugriff ist nur über einen erneuten Durchlauf möglich. Der Speicherbedarf bei diesem Parser-Typ ist geringer, weil nicht notwendigerweise das vollständige Dokument eingelesen werden muss.
- *Pull Parser*: Dabei wird ein kleiner Teil des Gesamtdokuments eingelesen. Die enthaltenen Daten können entsprechend weiterverarbeitet werden. Es folgt in einem weiteren Schritt das Parsen des nächsten Teils des Dokuments. Auch bei dieser Variante ist der Speicherbedarf geringer als bei einem Model Parser, da bei einem Schritt jeweils nur ein beschränkter Teil des Dokuments eingelesen wird.

Weil CDA-Dokumente in ihrem Aufbau komplex und variabel sind, zeigte sich, dass ihre Verarbeitung über einen Push- oder Pull-Parser aufwändig ist und sich die Anfragen an das Dokument zur Extraktion der relevanten Daten deshalb verschachtelt gestalten (Hillebrand 2006, 72). Weil die Umfänge der übermittelten CDA-Dokumente als überschaubar eingeschätzt werden (Hillebrand 2006, 72), kann für die Implementierung der mobilen Applikation trotz der beschränkten Ressourcen des Endgerätes ein Model-Parser gewählt werden. Damit sind insbesondere die Einschränkungen der anderen Parser-Typen nicht gegeben.

Im vorliegenden Fall wird als Basis für die Implementierung eines geeigneten CDA-Parsers der Model-Parser *Xparse-J*[63] eingesetzt. Dieser Parser ist in seiner Funktionalität hinsichtlich der Kompatibilität mit DOM zwar eingeschränkt (Hillebrand 2006, 43), benötigt deshalb aber keinen umfangreichen Speicherplatz zur Ausführung und eignet sich somit als Model Parser zur Ausführung auf einem mobilen Endgerät (Hillebrand 2006, 72).

---

[61] Ein Beispiel für einen häufig eingesetzten Model Parser in einer J2SE-Umgebung ist das Document Object Model (DOM, Ghosh 2003; Hillebrand 2006, 42). Weitere Informationen zum Document Object Model können auf den Seiten des W3C abgerufen werden: http://www.w3.org/DOM/, zugegriffen am 02.01.2007.

[62] Der SAX-Parser (Simple API for XML) ist ein weit verbreiteter Vertreter von Push Parsern (Gosh 2003; Hillebrand 2006, 42). Weitere Informationen zur Simple API for XML können den Projektseiten zu SAX entnommen werden: http://www.saxproject.org/, zugegriffen am 02.01.2007.

[63] Für weitere Informationen zur Xparse-J siehe http://www.webreference.com/xml/tools, zugegriffen am 02.10.2006.

In Abbildung 6.7-3 ist das Ergebnis der beschriebenen Architekturentscheidungen für die mobile Anwendung dargestellt. Dabei wird insbesondere die Modularisierung der gesamten Applikation in einer Schichtenarchitektur deutlich. Somit kann die prototypisch entwickelte Implementierung auch auf andere mobile Endgeräte übertragen werden, die über eine geeignete Java-Laufzeitumgebung verfügen.

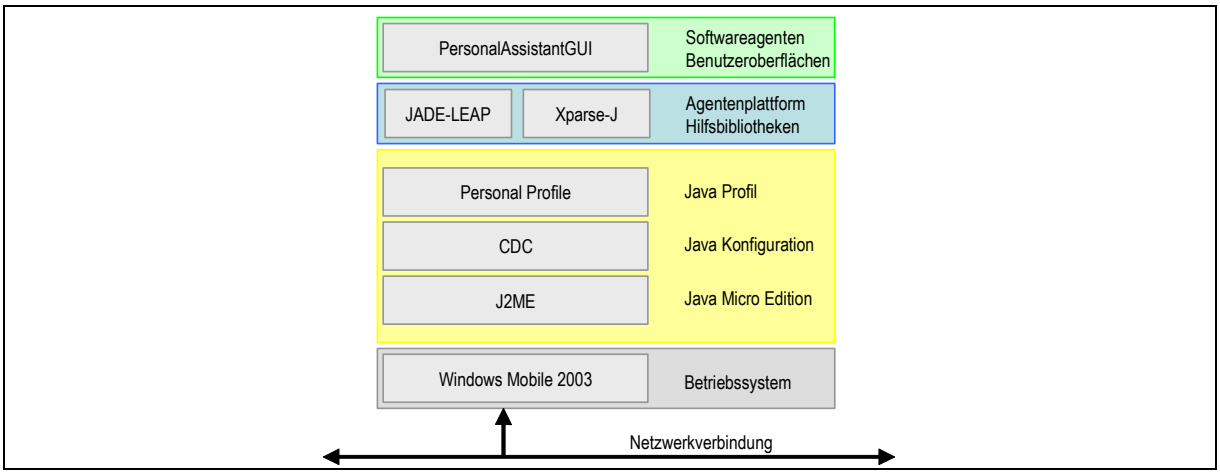

Abbildung 6.7-3: ***Schichtenarchitektur im Hinblick auf Architekturentscheidungen***
Quelle: *Hillebrand* (2006, 72) und *Schweiger et al.* (2006a, 40)

In Abbildung 6.7-4 ist das Ergebnis der Architekturentscheidungen hinsichtlich der Verteilung der Laufzeitumgebungen auf Rechner dargestellt. Wesentlich dabei ist, dass neben der JADE-Plattform eine separate JADE-LEAP-Plattform ergänzt wird, die wiederum in ihrem Split-Ausführungsmodus aufgeteilt ist. FrontEnd bzw. BackEnd laufen auf dem mobilen Gerät bzw. dem Desktop-Rechner. Der gemeinsame Split-Container erstreckt sich dabei auf die beiden genannten Knoten. Die Kommunikation innerhalb des verteilten Split-Containers erfolgt auf der Basis des JICP-Protokolls (siehe dazu Abschnitt 6.3.3). Auf dem BackEnd wird der Main Container ausgeführt, welcher die Management-Komponenten der JADE-Plattform enthält (Directory Facilitator (DF), Agent Management System (AMS), Remote Monitoring Agent (RMA), Hillebrand 2006, 73). Der Main Container übernimmt auch die hauptsächliche Netzwerkkommunikation (Caire 2006, 8), womit letztere auf dem mobilen Gerät Ressourcen schonend eingesetzt werden kann. Zudem kann auf diese Weise der Stromverbrauch des mobilen Endgerätes reduziert werden. Somit werden mit dieser Architektur die besonderen Gegebenheiten von mobilen Endgeräten unterstützt.

Weiterhin ist in Abbildung 6.7-4 die Desktop-Applikation mit dem wesentlichen Element AMD dargestellt. Die Kommunikation zwischen der JADE und dem BackEnd der JADE-LEAP-Plattform erfolgt über ein reguläres TCP/IP-Protokoll, das in der vorliegenden Arbeit drahtgebunden zur Verfügung gestellt wird.

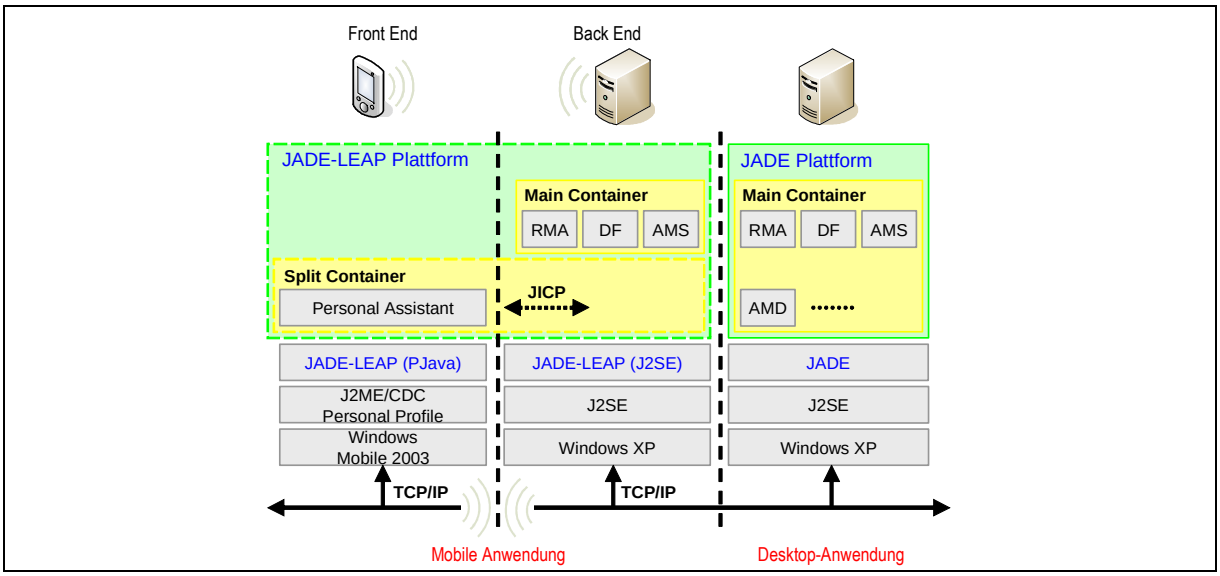

Abbildung 6.7-4: ***Detaillierte Darstellung der Schichten und ihrer Plattformen***
Quelle: *Hillebrand* (2006, 73) und *Schweiger et al.* (2006a, 41)

Die Ausführungen in diesem Abschnitt zeigen, dass die Architektur für eine agentenbasierte Anwendung auf mobilen Endgeräten entsprechend den vorhandenen Hardware-Ressourcen konfiguriert werden kann (Anforderung 14 aus Abschnitt 6.4.2). Damit wird insbesondere die Portierbarkeit der mobilen Anwendung auch auf andere mobile Endgeräte unterstützt.

6.7.4 Scheduling

6.7.4.1 Auswahl eines geeigneten Scheduling-Algorithmus

Grundlage für die Implementierung eines Scheduling-Algorithmus (Anforderung 10 aus Abschnitt 6.4.1) zur Realisierung einer dezentralen Terminplanung in einer Krankenhausumgebung ist die bei *Paulussen/Rothlauf/Heinzl* (2001) beschriebene Verarbeitungsvorschrift. Diese wird in einer agentenbasierten Implementierung in der vorliegenden Arbeit umgesetzt und in das Konstrukt des AMD bzw. in ein ausgewähltes KIS simuliert integriert. Für letztere Anforderung wird die Schnittstelle zum KIS entsprechend den Ausführungen in Abschnitt 6.6.5 und des Prototypenbaus (Abschnitt 1.8) simuliert.

Ausgangspunkt für die Wahl eines geeigneten Scheduling-Algorithmus sind die folgenden Charakteristika (Paulussen/Rothlauf/Heinzl 2001):

- Dezentrale Organisationsstruktur von Krankenhäusern durch eine Aufteilung in meist autonome Funktionsbereiche: In letzteren werden die jeweiligen Zeitpläne so erstellt, wie sie aus Sicht des jeweiligen Bereiches hinsichtlich der Ressourcenauslastung am günstigsten sind, was dazu führen kann, dass die Reduktion von Wartezeiten der Patienten nicht berücksichtigt wird.
- Erschwerend kommt hinzu, dass nicht antizipierbare Ereignisse, wie z.B. Notfälle, eine Veränderung des Gesundheitszustandes des Patienten oder ein ex ante noch nicht detailliert determinierter Behandlungsprozess, für die bereits abgeschlossene Planung einen erneuten Durchlauf der Terminplanung erfordern können.

Diese Anforderungen der Dezentralität sowie Flexibilität sind für die Gestaltung einer geeigneten Scheduling-Verarbeitungsvorschrift relevant: Verteilte Algorithmen besitzen im Gegensatz zu zentralen Verarbeitungsvorschriften den Vorteil, dass die jeweiligen Informationen für die Planungsaktivität nicht an zentrale Einheiten weiterzugeben sind, sondern in dezentralen Komponenten verwaltet werden können (Adam 1993, 471; Paulussen/Rothlauf/Heinzl 2001). Als Folge davon ist eine entsprechende Skalierung des Planungsalgorithmus möglich (Adam 1996; Paulussen/Rothlauf/Heinzl 2001), die insbesondere die dezentrale Organisationsstruktur in Krankenhäusern berücksichtigt.

Bei den Scheduling-Ansätzen von *Decker/Li* (1998) und *Schmidt/Urban* (1999) ist anzumerken, dass darin Ressourcen fokussiert und die Verhandlungen zwischen den Agenten nicht im Detail oder gänzlich nicht klassifiziert werden (Paulussen/Rothlauf/Heinzl 2001). In dem agentenbasierten Ansatz von *Vermeulen et al.* (2006) ist eine Berücksichtigung der Reihenfolgerestriktionen von Terminen sowie eines variierenden Gesundheitszustandes des Patienten nicht gegeben (Riedl 2007, 32-33). Diese Einschränkungen werden zwar durch die Arbeiten bei *Paulussen/Rothlauf/Heinzl* (2001) aufgehoben, dennoch fehlt dabei die Integration eines KISs bzw. in ein AMD für einen Patienten.

Aus den genannten Gründen wird in der vorliegenden Arbeit ein verteilter, agentenbasierter Ansatz gewählt, um die Anforderungen nach Dezentralität sowie Flexibilität zu erfüllen. Dazu wird der bei Paulussen/Rothlauf/Heinzl (2001) beschriebene Ansatz erweitert. In den folgenden Abschnitten 6.7.4.2 und 6.7.4.3 werden – aufbauend auf den Arbeiten bei Riedl (2007) – die wesentlichen Eigenschaften dieser Lösung dargestellt.

6.7.4.2 Architektur der Scheduling-Anwendung

Der für die in der vorliegenden Arbeit erstellte Modelltyp (siehe Abbildung 6.6-13) wird für die Terminplanung um den Typ SchedulingTaskAgent ergänzt (Riedl 2007, 48-49), indem dieser vom Typ TaskAgent abgeleitet wird. Weiterhin werden für das Scheduling von Patiententerminen (PatientSchedulingTaskAgent, für eine detaillierte Darstellung siehe Anhang A.5.14) bzw. Ressourcen-Terminen (ResourceSchedulingTaskAgent, für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.13) dedizierte Agenten eingeführt (Riedl 2007, 48-49), welche jeweils als Unterklasse der Klasse SchedulingTaskAgent realisiert sind. Entsprechend den Ausführungen bei *Paulussen/Rothlauf/Heinzl* (2001) repräsentieren diese Agenten jeweils einen Patienten bzw. eine Ressource. Diese Agenten besitzen ihren eigenen Terminplan sowie eine Zielfunktion, mit der eine Bewertung einer Planung bzw. ihrer Veränderung durch einen Koordinationsdurchlauf erfolgt.

Für eine Demonstration der implementierten Scheduling-Abläufe werden exemplarisch vier Ressourcen durch Ressourcen-Agenten gekapselt (siehe Abbildung 6.7-5): CT, Operationssaal, Röntgen und EKG. Um die Integration der Scheduling-Anwendung in ein KIS zu zeigen, werden die für die Terminplanung erforderlichen Informationen jeweils dem Informationssystem entnommen. Dazu werden die Anfragen an das KIS über den Agenten HISWrapper (siehe dazu Abschnitte 6.8.1 und A.5.4) mediiert. Weil zum gegenwärtigen Zeitpunkt keine geeigneten BAPIs für das gewählte KIS existieren, wird der Zugriff auf das KIS über adäquate Schnittstellen simuliert. Weiterhin ist die Scheduling-Funktionalität in das AMD eingebettet (siehe dazu die Ausführungen in Abschnitt 6.5.1).

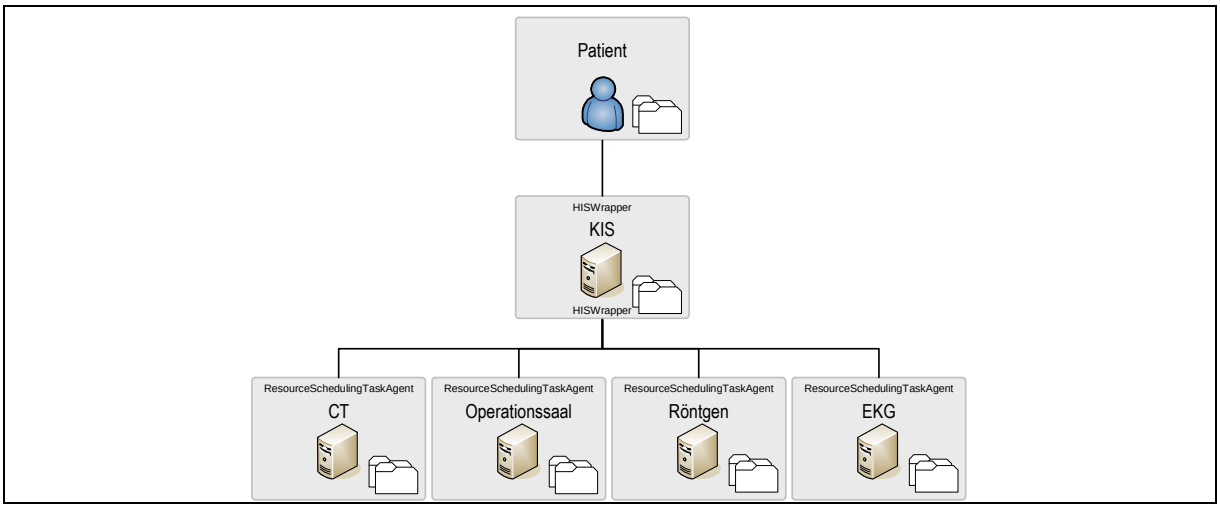

Abbildung 6.7-5: ***Architektur für die Scheduling-Anwendung***
Quelle: Eigene Darstellung

6.7.4.3 Beschreibung des Scheduling-Algorithmus

Aufbauend auf der in Abschnitt 6.7.4.2 beschriebenen Architektur werden im Folgenden die Charakteristika der Scheduling-Verarbeitungsvorschrift beschrieben. Dazu werden zunächst die erforderlichen Rahmenbedingungen dargestellt, bevor der eigentliche Algorithmus erläutert wird. Abgeschlossen wird dieser Abschnitt mit einer Darstellung der Eigenschaften der Terminierung, des Nichtdeterminismus, der Flexibilität, von Sonderfällen sowie von Einschränkungen, die zur Reduktion der Implementierungskomplexität vorgenommen werden.

Die grundlegenden Ideen umfassen nach *Paulussen/Rothlauf/Heinzl* die folgenden Eigenschaften (2001):

Ressourcen-Agenten streben die Erhöhung der Auslastung ihrer Ressource an. Patientenagenten hingegen besitzen das Ziel, die Durchlaufzeit für eine Behandlung und damit die Wartezeit für Termine sowie in der Summe die gesamte Behandlungsdauer zu reduzieren. Dies hat zur Folge, dass die Patientenagenten um die beschränkten Ressourcen konkurrieren. Die Aufgabe der Agenten ist es nun, durch Verhandlungen den Wert ihrer Zielfunktion zu maximieren. Bei der Maximierung der Zielfunktion ist jedoch zu beachten, dass nach marktwirtschaftlichen Gesichtspunkten ein Zustand des Gesamtsystems mit der Eigenschaft der Pareto-Effizienz (Varian 1994, 226-227 und 332-336) dann erreicht wird, wenn kein Agent seine Zielfunktion zu Lasten eines anderen Agenten weiter erhöhen kann. Die dafür erforderliche Kostenfunktion wird bei *Paulussen/Rothlauf/Heinzl* über die folgenden Überlegungen hergeleitet (2001):

Da das Ziel der Patientenagenten die Reduktion der Durchlaufzeit ist, ist eine entsprechende Funktion zu wählen, die einen höheren absoluten Wert liefert, je länger die Behandlung dauert. Für diese Anforderung eignet sich eine Exponentialfunktion, bei der die Kosten für eine Behandlung in einer späteren Zeiteinheit die Summe aller bisherigen Kosten für die Behandlungen in den vorhergehenden Zeiteinheiten übersteigen (Bowman 1959, 623). Die Grundform der Kostenfunktion besitzt deshalb die folgende Definition:

$$k(t) = g^t$$

Dabei bezeichnet g die Priorität des Patienten und t die Zeit, die zwischen der Ankunft des Patienten und dem vorgesehenen Beginn des Termins liegt.

Damit der minimale Gesamtbehandlungszeitraum zwischen allen Patienten vergleichbar wird, erfolgt eine Normierung des Startzeitpunktes für eine Behandlung:

$$t^{Norm} = -\delta^{Gesamt} + t - 1$$

δ^{Gesamt} spezifiziert die minimale Gesamtdurchlaufzeit und t den tatsächlichen Zeitpunkt für den Beginn eines Termins. Zur Vereinfachung wird im Folgenden statt t^{Norm} nur t referenziert.

Die Ermittlung der eigentlichen Kosten für eine einzelne Behandlung erfolgt über eine Integralberechnung:

$$K(t,\delta) = \int_t^{t+\delta} g^t = \frac{1}{\ln g}\left(g^{t+\delta} - g^t\right)$$

Die Kosten für die Gesamtbehandlung berechnen sich aus der Summe der Kosten für die Einzelbehandlungen. Wird nun eine Behandlung aufgrund einer Terminverschiebung auf der Zeitachse nach hinten bzw. vorne verschoben, werden die Kosten erhöht bzw. reduziert. Damit kann dieser Effekt durch die Differenz der beiden genannten Zeitintervalle mit folgenden Kosten belegt werden:

$$p(t,t^{neu},\delta) = \int_t^{t+\delta} g^t - \int_{t^{neu}}^{t^{neu}+\delta} g^{t^{neu}} \text{ bzw. } p(t,\Delta,\delta) = \frac{1}{\ln g} g^t\left(1 - g^{-\Delta}\right)\left(-1 + g^{\delta}\right), \text{ wobei } \Delta = t - t^{neu}$$

Dabei bezeichnet t den ursprünglichen Zeitpunkt für den Termin, t^{neu} den Zeitpunkt nach der Verschiebung des Termins und δ die Dauer für die Einzelbehandlung. Auf der Grundlage dieser Zielfunktion können nun Patientenagenten determinieren, welche Kosten bei einem Termintausch mit einer zeitlichen Verlängerung mindestens von einem Nachfrager zu fordern sind, bzw. wie hoch der Betrag ist, um eine Verkürzung der Behandlungsdauer zu erzielen, ohne nach dem Termintausch eine schlechtere Position einzunehmen als vorher. Analog zu den Überlegungen für die Wahl der Kostenfunktion für Patienten werden diese ohne weitere Änderung auch auf Ressourcen-Agenten übertragen, denn deren Ziel ist ebenso die Reduktion von Leerlaufzeiten, was einer zeitlichen Verschiebung von Terminen auf dem Zeitstrahl nach vorne entspricht.

Außer der beschriebenen Zielfunktion ist für den Scheduling-Algorithmus relevant, dass Einschränkungen bezüglich der Reihenfolge von Terminen berücksichtigt werden. Diese Restriktionen werden durch Nebenbedingungen über die eingesetzten Zielfunktionen abgebildet.

Außerdem sind vom Scheduling-Algorithmus auch solche Termine zu berücksichtigen, die unabhängig von der Präsenz des Patienten stattfinden und damit nebenläufig ausgeführt werden können. Diese patientenunabhängigen Termine stehen im Gegensatz zu patientenabhängigen Terminen, bei denen Patienten anwesend sein müssen und die deshalb nicht parallel ausgeführt werden können.

Aufbauend auf diesen Eigenschaften wird die grundlegende Funktionsweise des Scheduling-Algorithmus von *Paulussen/Rothlauf/Heinzl* wie folgt beschrieben (2001): Patientenagenten versuchen, ihre Kostenfunktion zu minimieren, indem sie ihre Behandlungen auf der Zeitachse weiter nach vorne verlegen. Ist das von einem Agenten angestrebte Zeitintervall bereits von einem anderen Agenten belegt, kann der Agent versuchen, mit der durch die Terminverschiebung induzierten Kostenersparnis einen Termin von einem anderen Agenten zu erwerben. Damit ein solcher Agent willig ist, seinen ursprünglichen Termin aufzugeben, müssen die damit assoziierten Kosten geringer sein als der Kostenvorteil des Käufers für den Termin. Auf der Basis dieser Überlegungen wird im Folgenden der Scheduling-Algorithmus nach *Paulussen/Rothlauf/Heinzl* (2001) beschrieben:

1. Ein Patientenagent bestimmt als Nachfrager, eingeschränkt durch eine evtl. vorgegebene sequenzielle Reihenfolge von Terminen, denjenigen Termin, durch den er durch eine Verschiebung möglichst weit nach vorne die umfangreichste Ersparnis erhält.
2. Nach der Identifikation eines geeigneten Termins beginnt der Nachfrager den Verhandlungsalgorithmus um den ausgewählten Termin. Dazu informiert der Patientenagent den entsprechenden Ressourcen-Agent über den ursprünglichen sowie den potenziellen Termin.
3. Für die Verhandlung wird der zur Disposition stehende Termin vom Ressourcen-Agenten entsprechend klassifiziert, der angefragte Termin wird als belegt gekennzeichnet.
4. Der Ressourcen-Agent informiert alle von der Terminverschiebung betroffenen, als Anbieter fungierenden Agenten. Können von der Ressource Patienten nebenläufig bedient werden, werden alle alternativen Ressourcen-Agenten darüber informiert, ein Angebot für den angefragten Termin abzugeben.
5. Die Termine anbietenden Patientenagenten berechnen auf der Grundlage der beschriebenen Kostenfunktion die Kosten, die durch die Terminverschiebung induziert würden, und leiten diese Information an den Initiator der Terminänderung weiter. Diese Agenten verhandeln nun mit der Ressource in der Rolle als Nachfrager über den angebotenen Alternativtermin. Ziel ist dabei, die Kosten so gering wie möglich zu halten. Dabei können die Patientenagenten letztendlich einen späteren Termin als ihren ursprünglichen erhalten, wenn ihre dadurch entstehenden Kosten durch den Kostenvorteil des ursprünglichen Anfragers mindestens ausgeglichen werden können.
6. Sind kaskadierende Terminverschiebungen durch Reihenfolgerestriktionen erforderlich, sind auch diese Kosten von dem ursprünglich anfragenden Agenten auszugleichen.
7. Wenn für den ursprünglichen Patiententermin des Nachfragers eine Überschneidung bei einer anderen Ressource gegeben ist, werden von diesem Patientenagenten in der Rolle des Anbieters die Kosten für eine Terminverschiebung in Erfahrung gebracht.
8. Ist der für die in der Rolle der Anbieter fungierenden Agenten gewünschte Termin bereits reserviert, werden die Schritte 2 mit 7 als Nachfrager für diese Agenten ausgeführt.

Wenn dieser Termin hingegen belegbar ist, können die Kosten für die Verschiebung berechnet und an den Nachfrager weitergegeben werden. Fungiert dieser Nachfrager auch in der Rolle eines Anbieters, können auch seine Kosten dem Nachfrager der Terminverschiebung übermittelt werden. Diese Abarbeitung wird so lange rekursiv durchlaufen, bis der Anbieter mit dem die Terminverschiebung initiierenden Agenten in direkter Interaktion steht und die Kosten weitergibt. Die Kosten für die gesamte Terminverschiebung ergeben sich damit als Summe aller erforderlichen Änderungen.

9. Wenn der Nutzen die Kosten für die Terminverschiebung beim Initiator übersteigt, wird dem entsprechenden Ressourcen-Agenten die tatsächliche Verschiebung mitgeteilt. Die temporär vorgenommenen Freigaben und Reservierungen von Terminen werden festgeschrieben. Andernfalls, d.h. der Nutzen übersteigt nicht die Kosten, werden die vorübergehend reservierten bzw. freigegebenen Termine wieder als frei bzw. belegt gekennzeichnet. Wenn eine Ressource Patienten nebenläufig behandeln kann, wird nur das kostengünstigste Angebot eines Patientenagenten für die weitere Planung zugrunde gelegt.

Der beschriebene Algorithmus terminiert nach *Paulussen/Rothlauf/Heinzl* (2001), da dieser nicht in einem lokalen Optimum oder Suboptimum verbleibt, weil in diesem Zustand weiterhin Veränderungen der Terminpläne möglich sind.

Weiterhin ist anzumerken, dass der Algorithmus Nichtdeterminismus aufweisen kann, weil die Terminverhandlungen durch Agenten per se nebenläufig durchgeführt werden. So wird in dem Fall, dass mehrere Agenten gleichzeitig eine Terminverschiebung bei einer Ressource anfordern, die Weiterleitung der Nachrichten vom Rahmenwerk JADE gesteuert und unterliegt somit einer aus Sicht des Entwicklers nicht deterministischen Verarbeitung.

Der gewählte Algorithmus erlaubt außerdem die Reaktion auf nicht antizipierbare Ereignisse, indem bei einer Änderung des Gesundheitszustandes des Patienten seine Priorität entsprechend angepasst wird und dieser damit bei der Terminverschiebung höhere Gewinne erzielen kann (Paulussen/Rothlauf/Heinzl 2001). Mit dieser Eigenschaft wird die Flexibilität deutlich, die durch ein agentenbasiertes System gegeben ist.

Werden zusätzliche Behandlungen eingeplant, werden diese kostenneutral für das nächste Zeitintervall ohne Konflikte geplant (Paulussen/Rothlauf/Heinzl 2001). Steht eine Ressource z.B. aus technischen Gründen temporär nicht mehr zur Verfügung, werden die Agenten ebenso kostenneutral für den nächsten möglichen Termin eingeplant (Paulussen/Rothlauf/Heinzl 2001). Dieser Status der Terminplanung kann als Ausgangspunkt für Terminverschiebungen fungieren (Paulussen/Rothlauf/Heinzl 2001). Dazu wird der beschriebene Algorithmus erneut durchlaufen.

Zur Reduktion der Implementierungskomplexität werden die folgenden Einschränkungen bei der Umsetzung des Algorithmus gegenüber der Beschreibung bei *Paulussen/Rothlauf/Heinzl* (2001) vorgenommen, ohne jedoch die Bedeutung der Ergebnisse einzuschränken:

- Der Ausgangspunkt für ein Scheduling-Szenario ist die Annahme einer optimal vorliegenden Terminplanung. Eine Terminänderung wird erst dann angestoßen, wenn dies durch äußere Einflüsse, wie z.B. durch die nicht antizipierbare Verlängerung von Ter-

minen, erforderlich wird. Damit werden einerseits Ressourcen eingespart, weil nicht permanent Scheduling-Versuche unternommen werden. Andererseits entspricht dieser Ansatz den Bedingungen in der Domäne, in der Ereignisse auftreten, die eine Terminänderung erfordern. Mit diesen Annahmen vereinfacht sich der Algorithmus.

- Auf patientenungebundene Termine wird verzichtet, weil diese ohnehin keine Konflikte bei nebenläufiger Ausführung hervorrufen (Paulussen/Rothlauf/Heinzl 2001).
- Auf eine parallele Behandlung von Patienten in einer Ressource wird verzichtet, wodurch sich der Algorithmus weiter vereinfacht (siehe dazu Schritt 4 der Scheduling-Verarbeitungsvorschrift). Eine Einschränkung hinsichtlich der Qualität der Ergebnisse des Scheduling-Algorithmus ist nicht gegeben, weil bei einer Terminverschiebung bei einer Ressource potenziell ebenso mehrere Agenten beteiligt sein können.
- Neue Termine werden jeweils am Ende des Terminplans für den Agenten bzw. die Ressource ohne Kosten eingefügt. Die Reduktion der Leerlaufzeiten bzw. Wartezeiten wird von diesem Zustand des Scheduling-Systems angestoßen. Damit vereinfacht sich der Einfügealgorithmus, weil kein Intervall zwischen anderen Untersuchungsterminen identifiziert werden muss, welches der Anforderung der benötigten Intervalllänge genügen würde. Die Qualität des Algorithmus wird damit nicht reduziert, weil die potenziellen Terminänderungen nach dem bekannten Muster angestoßen werden und nur die Ausgangsposition verschieden ist.
- Um die Komplexität des Scheduling-Algorithmus weiter zu reduzieren, wird in der umgesetzten Implementierung auf die Berücksichtigung der Kostenfunktion für die Ressourcen verzichtet. Letztere werden dazu als passive Elemente umgesetzt, die ihre Terminpläne nicht zu verbessern versuchen. Da das Design so gestaltet ist, dass es hinsichtlich der Realisierung bei Patienten und Ressourcen symmetrisch ist, kann die Implementierung des Scheduling-Algorithmus auch auf Ressourcen übertragen werden.
- Konsequenzen bei einer Verbesserung des Gesundheitszustandes des Patienten bzw. bei Verkürzungen von Terminen werden bei dem Scheduling-Algorithmus von *Paulussen/Rothlauf/Heinzl* (2001) nicht ausgeführt. In dem Fall, dass sich der Gesundheitszustand des Patienten verbessert, muss zwar keine zwingende Terminverschiebung beim betreffenden Patientenagenten stattfinden, denn es entsteht keine Konflikt behaftete Situation. Dennoch ist aber das Potenzial für Patientenagenten gegeben, ihre Termine ggf. anders einzuplanen. Für eine entsprechende Umsetzung müsste der Scheduling-Algorithmus bei einer Verbesserung des Gesundheitszustandes bzw. auch bei einer Terminverkürzung für jeden Patientenagenten gestartet werden. Dazu müssten von jeder Ressource alle mit ihr assoziierten Patientenagenten informiert werden. Da der Scheduling-Algorithmus derart gestaltet ist, dass zu einem Zeitpunkt nur von einem Patientenagenten eine Umplanung durchgeführt werden kann, müsste das Scheduling der Termine koordiniert werden, da Patienten bei mehreren Funktionsbereichen Termine besitzen können. Diese Funktionalität eines Scheduling-Algorithmus bedarf einer weitergehenden Betrachtung, um eine geeignete Lösung zu identifizieren, und ist deshalb nicht Gegenstand der vorliegenden Arbeit.

Zusammenfassend kann festgestellt werden, dass mit dem beschriebenen Algorithmus die Reduktion der Leerlaufzeiten von Ressourcen sowie von Wartezeiten und somit auch die Gesamtdurchlaufzeit eines Patienten reduziert werden können (Paulussen/Rothlauf/Heinzl 2001). Offen bleibt bei dem beschriebenen Algorithmus die möglichst zeitnahe Aggregation von Terminen, um die Wegezeiten für Patienten zu reduzieren (Paulussen/Rothlauf/Heinzl 2001). Als Defizit können bei dem Ansatz von Paulussen/Rothlauf/Heinzl (2001) weiterhin die Integration der Planungsergebnisse in ein KIS sowie die Einbettung der Funktionalitäten in eine Patientenakte bzw. in das in der vorliegenden Arbeit beschriebene AMD festgestellt werden. Dazu wurden bereits in Abschnitt

6.7.4.2 entsprechende Architekturvorschläge beschrieben. Obwohl in der vorliegenden Arbeit zur Reduktion der Implementierungskomplexität Einschränkungen der Verarbeitungsvorschrift von Paulussen/Rothlauf/Heinzl (2001) vorgenommen werden, wird dadurch die Qualität der erzielten Ergebnisse nicht eingeschränkt.

6.7.5 Patientenportal

6.7.5.1 Überblick

Für eine geeignete Patientenorientierung wird die Bereitstellung eines Web basierten Portals umgesetzt (Anforderung 11 aus Abschnitt 6.4.1, Anforderung 17 aus Abschnitt 6.4.2). Dazu wird im Folgenden zunächst der Bedarf für eine solche Plattform abgeleitet. Es folgt eine Darstellung der Vorteilhaftigkeit von Web basierten Anwendungen und technischer Implementierungseigenschaften. Abschließend werden Einschränkungen der Implementierung beschrieben.

Obwohl Studien zu dem Ergebnis kommen, dass bisherige Plattformen technisch noch zu wenig ausgereift sind, wird Patientenportalen dennoch ein wesentlicher Stellenwert für die Integration des Patienten in den Behandlungsprozess beigemessen (für weitere Ausführungen dazu siehe Abschnitt 4.9.2). Deshalb wird in der vorliegenden Arbeit als Kompromiss vorgeschlagen, den Patienten über ein geeignetes Portal zwar in die Versorgung zu integrieren, indem er Einsicht in seine Gesundheitsdaten nehmen kann, aber dabei nur solche Einträge des Patienten zuzulassen, die keinen wesentlichen Einfluss auf die Qualität der Entscheidungen der Leistungserbringer besitzen. Diese Einträge werden in der vorliegenden Arbeit lediglich in einer lokalen Datenbank abgelegt, um die Implementierungskomplexität zu reduzieren. Für eine Einbettung in die TI wird in Analogie zu anderen selbst erhobenen Daten des Patienten (Bernnat/Booz Allen Hamilton GmbH 2006, 262) vorgeschlagen, diese Daten im Patientenfach der eGK abzulegen, auf das z.B. über die eKiosk-Funktionalität oder den HBA in Zusammenwirken mit der eGK zugegriffen werden kann.

Web basierte Anwendungen eignen sich für diese Integration des Patienten deshalb besonders, weil diese auf der Seite des Clients nur die Erfüllung von wenigen Anforderungen voraussetzen: Der Installationsaufwand für den erforderlichen Browser ist gering und die Anwendungslogik befindet sich auf der Server-Seite. Zudem können mit diesem Ansatz unterschiedliche Endgeräte und Hardware-Plattformen unterstützt werden. Damit können, zusammen mit den bisher beschriebenen Eigenschaften der prototypischen Implementierung, die bei *Bernnat* (2006, 82-83) angenommenen Anforderungen zur Realisierung der ausgewählten eKiosk-Funktionalität zur Einsichtnahme der elektronischen Patientenakte (Anforderung 11 aus Abschnitt 6.4.1) erfüllt werden. Eine SMC wird in der vorliegenden Arbeit insofern berücksichtigt, als die Funktionalität der Implementierung so gestaltet ist, dass beim Einsatz der Portalfunktionalität ein nicht wechselbarer HBA eingelegt wird. Diese Einschränkung zur Reduktion der Implementierungskomplexität ist gerechtfertigt, weil die SMC bei der eKiosk-Funktionalität für den Anwender transparent ist.

Bei der in dieser Arbeit beschriebenen Anbindung der virtuellen Patientenakte an das Portal wird aus den folgenden Gründen eine Web Services (siehe Abschnitt 3.2.4.4) basierte Lösung gewählt: Die Implementierung soll etablierte Standards unterstützen, um eine breite Unterstützung und Weiterentwicklung der Applikation zu ermöglichen. Weiterhin können zwischen

Web Services und einer agentenorientierten Anwendung Ähnlichkeiten hinsichtlich der Bereitstellung von Diensten identifiziert werden. Eine detaillierte technische Umsetzung der Transformation zwischen Web Services sowie Agentendiensten und umgekehrt ist in Abschnitt 6.3.5 beschrieben.

Die Portaltechnologie wird als Grundlage für die Realisierung des Patientenportals in der vorliegenden Arbeit ausgewählt, weil Portale[64] per se Daten aus verschiedenen Informationssystemen zusammenstellen. Weiterhin kann die jeweils visualisierte Information an die Anforderungen der Nutzer angepasst werden. Somit werden unterschiedliche Sichten auf Daten realisiert. Dieses Konzept ist für eine Anwendung im Gesundheitswesen besonders geeignet, weil auch hier unterschiedliche Sichten für Patienten gewünscht sind. Weitergehende Ausführungen zur Vorteilhaftigkeit der Portaltechnologie sind bei *Taranovych et al.* (2006, 238-240) beschrieben.

Um bereits implementierte Grundlagen eines Portals (Taranovych et al. 2007) wiederverwenden und so den Implementierungsaufwand reduzieren zu können, wird in der vorliegenden Arbeit das Open Source Portalsystem Liferay[65] gewählt. Für den Einsatz dieser Implementierung spricht die vollständige Kompatibilität zum für Portlets wesentlichen Standard JSR (Java Specification Request) 168[66]. Portlets, die diesem Standard folgen, können in Portalen deployt werden, die kompatibel zu diesem Standard sind. Somit wird die Idee der Plattformunabhängigkeit der Sprache Java auf Portlets übertragen, die in jeder kompatiblen Umgebung lauffähig sind. Außerdem können mit Liferay auch mobile Endgeräte die Benutzerschnittstelle zur Verfügung stellen.

Da die grundsätzliche Integration von verteilten Informationssystemen in die virtuelle Patientenakte in der vorliegenden Arbeit bereits demonstriert wurde (siehe dazu die Abschnitte 6.7.1 und 6.8.1), wird auf die Anpassung der agentenbasierten Anwendung zur Aggregation der durch den Patienten selbst eingegebenen Daten des Patientenportals verzichtet. Da diese Daten in einer lokalen Datenbank abgelegt werden, würde damit nur eine geeignete Schnittstelle zwischen der Datenbank und der Anwendung entsprechend dem Wrapper-Ansatz (Jennings 2001, 40) umzusetzen sein. Damit würde aber kein weiterer Mehrwert für die vorliegende Arbeit erzielt werden. Deshalb werden im Folgenden nur die Aggregation von Patientendaten

---

[64] Ein Portal kann als eine Web basierte Anwendung betrachtet werden, die die Eigenschaften Personalisierung, einmalige Anmeldung am System, Aggregation von Inhalten aus unterschiedlichen Quellen sowie Visualisierung eines Informationssystems umfasst (Linwood/Minter 2004, 1) und eine Menge von Portlets zusammenfasst (Eberhardt/Gurzki/Hinderer 2002, 13). Damit ist zusätzlich die Eigenschaft verbunden, „ortsunabhängig einen zentralen Zugang zu Ressourcen“ (Sun Microsystems Inc. 2003, 13) zur Verfügung zu stellen. In einem Portlet innerhalb eines Portals werden die Informationen vergleichbar wie in einem Fenster einer Desktop-Anwendung inhaltlich zusammenhängend dargestellt. Ein Portlet ist eine Web basierte Java-Komponente, die Anfragen verarbeitet und dynamischen Inhalt an der Benutzeroberfläche visualisiert (Linwood/Minter 2004, 1). Die Benutzerschnittstelle wird über einen Browser angezeigt, womit Portale auf unterschiedlichen Endgeräten und Plattformen verfügbar gemacht werden können (Taranovych et al. 2006, 238-240 und 252). Für weitere Ausführungen zur Portaltechnologie und ihrer Vorteilhaftigkeit aus technischer Perspektive wird auf *Taranovych et al.* (2007) verwiesen.

[65] Weitere Informationen zu Liferay können auf den folgenden Seiten abgerufen werden: http://www.liferay.com/web/guest/home, zugegriffen am 10.01.2007.

[66] Weitere Informationen zum Java Specification Request 168 können auf den folgenden Seiten abgerufen werden: http://jcp.org/en/jsr/detail?id=168, zugegriffen am 10.01.2007.

aus Primärsystemen der Leistungserbringer und ihre Visualisierung in einem Patientenportal berücksichtigt.

6.7.5.2 Architektur

Entsprechend der bei *Waegemann* (1999, 116) eingeführten Integrationsstufe der „Electronic Health Record“ wird in der beschriebenen Implementierung die Eingabe von eigenen Daten durch den Patienten sowie die Aggregation von bisherigen Daten aus der virtuellen Patientenakte unterstützt. Um das Portalsystem mit der Agentenplattform zu koppeln, wird die WSIG-Komponente (siehe dazu Abschnitt 6.3.5, Anforderung 11 aus Abschnitt 6.4.1, Anforderung 19 aus Abschnitt 6.4.2) eingesetzt. Dazu erfolgt zwischen dem Portalsystem und der Agentenplattform über die WSIG-Schnittstelle eine synchrone Kommunikation (siehe Abbildung 6.7-6).

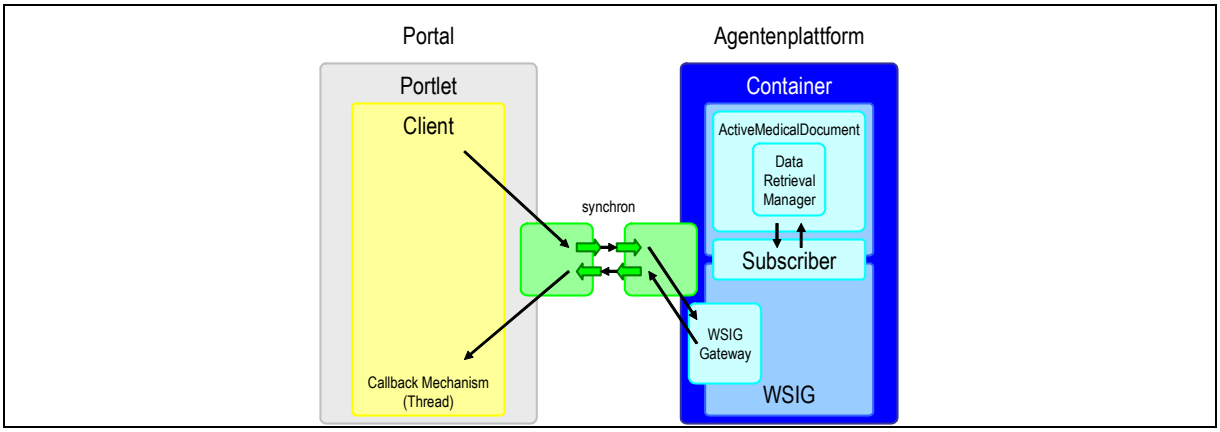

Abbildung 6.7-6: ***Kopplung zwischen der virtuellen Patientenakte und dem Portalsystem***
Quelle: In Anlehnung an *Michelbach/Druker* (2006, 55)

Die WSIG-Erweiterung für JADE übersetzt dabei Aufrufe eines Clients über Web Services in ACL-Nachrichten und vice versa. An dieser Stelle zeigt sich die Vorteilhaftigkeit von wohl definierten Eintrittspunkten in die virtuelle Patientenakte. Damit wird der Kommunikationsaufwand zwischen der Agenten- und der Web Services-Anwendung bei der Aggregation der Patientendaten minimiert. Dies reduziert den Aufwand für die Transformation der ausgetauschten Nachrichten.

Die Komponente Subscriber (siehe Abbildung 6.7-6) mediiert zwischen der Agentenplattform und der WSIG-Erweiterung. Die Ergebnisse der Informationsaggregation in der Agentenanwendung werden über die WSIG-Komponente von einer ACL-Nachricht in eine Rückantwort an den initiierenden Web Services-Aufruf transformiert. Im Portal erfolgt die weitere Verarbeitung dieser Antwort, um schließlich die aggregierten Patientendaten zu visualisieren.

6.7.6 Agentenbasierte Prozessunterstützung

6.7.6.1 Überblick

Die Grundlage für die Realisierung der Unterstützung von interaktiven Behandlungsprozessen bildet das von *Meyer* (2004) entworfene Prozessmanagement in MASen. Um komplexe Behaviours für Agenten zu erhalten und damit Prozesse geeignet zu unterstützen, können zusammengesetzte Behaviours konstruiert werden, für die das JADE-Rahmenwerk geeignete Behaviour-Klassen zur Verfügung stellt. *Meyer* identifiziert bei dieser Herangehensweise jedoch wesentliche Defizite (2004, 1-2):

- Die Komposition von beliebigen Behaviours zu einem Gesamtverhalten eines Agenten ist problematisch, weil der Transfer von Daten zwischen verschiedenen Teilverhalten nicht spezifiziert ist.
- Diese Komposition von Behaviours erfolgt auf der Basis von Programmtext, womit eine dynamische Änderung des Verhaltens zur Laufzeit erschwert wird.

Um diese Probleme zu lösen, schlägt *Meyer* (2004) die Beschreibung des Verhaltens in Form von Prozessen vor. Dabei werden die folgenden Eigenschaften angestrebt (Meyer 2004, 1-2):

- Die Prozessbeschreibungen sind komposit und können damit mehrere Teilprozesse und atomare Aktivitäten enthalten.
- Atomare Aktivitäten werden durch Behaviours in einer Agentenplattform realisiert.
- Mit diesem Ansatz ist es möglich, zur Laufzeit beliebige Prozesse zu komponieren oder die Ausführungsreihenfolge der Aktivitäten dynamisch zu ändern.
- Unterstützung der in Agentensystemen herausragenden Eigenschaft der Flexibilität: Das für den Agenten adäquate Verhalten kann zur Laufzeit angepasst werden.

Um diese Anforderungen an eine Prozessbeschreibung zu erfüllen und eine flexible Integration von Informationssystemen (Zang/Hofer/Otmar 2004, 484) auf der Basis eines Prozesses aus Patientensicht zu unterstützen, ist die Definition eines geeigneten Datenmodells erforderlich. Letzteres orientiert sich an internationalen Standards, die bspw. von der *Object Management Group* (2007a), der *Workflow Management Coalition* (WfMC 2005) oder durch ebXML (Electronic Business using eXtensible Markup Language, Organization for the Advancement of Structured Information Standards 2006) und WSBPEL (OASIS 2007) definiert werden, ohne jedoch den vollständigen Spezifikationsumfang, sondern nur die für die Anforderungen relevanten Teile zu berücksichtigen (Krcmar/Horn 2003, 23), und wird in Form eines Metamodells für Prozesse in MASen (siehe auch Abbildung 6.7-7) wie folgt beschrieben (Meyer 2004, 9 und 15-17):

- Ein Prozess (Process) ist komposit und setzt sich aus weiteren Teilprozessen und atomaren Aktivitäten (Activity) zusammen.
- Die Teilprozesse und Aktivitäten werden parallel abgearbeitet. Sequenzialisierungen werden durch Constraints definiert. Constraints können mit disjunktiven oder konjunktiven Konnektoren verknüpft sein. Damit lassen sich die Prozessmuster Sequence, Parallel Split, Synchronization, Simple Merge, Multiple Choice, Discriminator und Implicit Termination modellieren.

- Definition der Constraints über die Daten (Artifact), die Prozesse generieren oder verändern, und Zusammenfassung von Constraints und Artifacts zu einer Klasse Repository
- Die Ein- und Ausgabe von Daten in bzw. an Aktivitäten wird durch Parameterassoziationen realisiert (ParameterAssociation). Diese Parameter werden den Aktivitäten übergeben und sind den jeweiligen Prozessen zugeordnet.
- Die Interagentenkommunikation wird durch zwei spezialisierte Aktivitäten, SendMessageActivity und ReceiveMessageActivity realisiert.
- Abbildung der Aufgabendelegation über eine Relation zwischen einem Prozess und einem Agenten (delegatedTo)

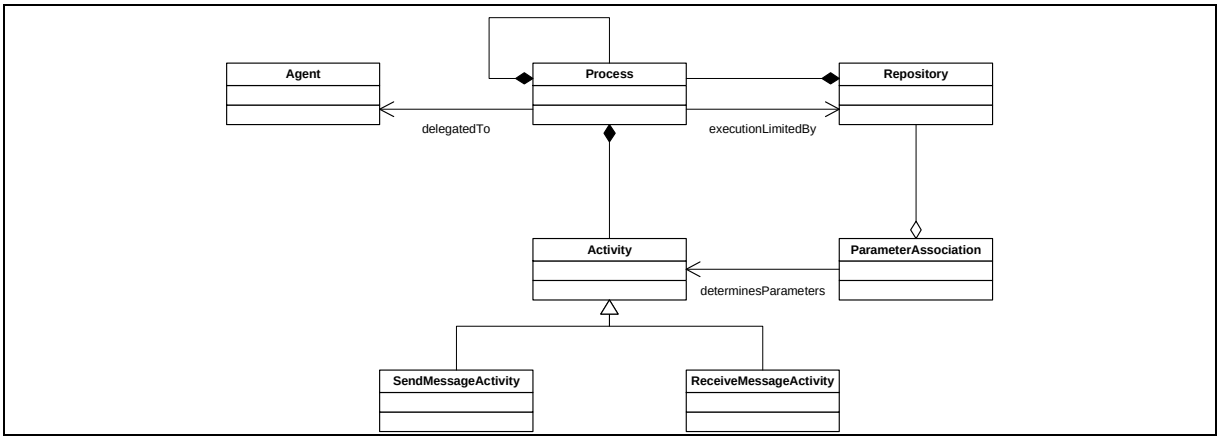

Abbildung 6.7-7: ***Metamodell für Prozesse in Multiagentensystemen***
Quelle: *Meyer* (2004, 17)

Dieses Metamodell (PML, Process Markup Language) zur Beschreibung der Prozessstruktur (siehe Abbildung 6.7-8) wird in einer XML-Beschreibung abgebildet, die die folgenden Elemente besitzt (Meyer 2004, 18-22; Bergmann/Klinger/Winkler 2006, 16-21):

- *process*: Definition eines Prozessschrittes
- *activity*: Spezifikation des Behaviours, das vom Agenten ausgeführt werden soll
- *parameter*: Spezifikation eines Parameters über den Typ value (Übergabe eines Wertes) oder den Typ binding (Ergebnis einer Aktivität)
- *constraints*: Spezifikation der Vorbedingen, die zu erfüllen sind, wenn die assoziierte Aktivität ausgeführt werden soll. Die constraints können durch einen disjunktiven oder konjunktiven Konnektor verknüpft werden. Beide Typen können aber nicht gleichzeitig für eine constraint-Menge definiert werden. Solche constraints enthalten als Attribute einen Namen, der mit einer Aktivität assoziiert ist, und den Wert des Rückgabewertes dieser Aktivität.
- *agent*: Spezifikation des Agenten, der das im activity-Element spezifizierte Behaviour ausführt. Dieses Element besitzt die Attribute type (Klassenname) und name (registrierter Name).
- *fipa-message*: Beschreibung einer vollständigen FIPA-Nachricht
- *service-description*: Angabe eines Dienstes über die Spezifikation eines AID

```
<Prozess> ::=                 "<process name=>" <String> ">"
                                                    (<Constraints>)*
                                                    (<DelegierterAgent>){0,1}
                                                    (<Aktivitaet>)*
                                                    (<Prozess>)*
                              "</process>"

<Constraints> ::=             "<constraints type=" ("and" | "or") ">"
                                                    (<Constraint>)*
                              "</constraints>"

<Constraint> ::=              "<constraint name=" <Aktivitaetsname> ">"
                                                    <RegEx>
                              </constraint>"

<DelegierterAgent> ::=        "<agent type=" <Klassenname> "name" <String> ">"
                              </agent>"

<Aktivitaet> ::=              "<activity type=" <Klassenname> "name=" <String> ">"
                                                    (<Parameter>)*
                                                    (<Constraints>)*
                              </activity>"

<Parameter> ::=               "<parameter name=" <String> (
                                                    "type=value" <String> |
                                                    "type=binding" <Aktivitaetsname>)
                                                    (<FIPA-Nachricht>)*
                                                    (<Dienstbeschreibung>)*
                              "</parameter>"

<Aktivitaetsname> ::=         ("::"){0,1} <String> ("::" <String>)*

<Klassenname> ::=             <String> ("." <String>)*

<FIPA-Nachricht> ::=          "<fipa-message act=" <Performativ> ">"
                                                    (<Inhalt>)
                                                    (<Empfaenger>)
                                                    (<Protokoll>)
                                                    (<Sprache>)
                                                    (<Ontologie>)
                              "</fipa-message>"

<Performativ> ::=             {Performative}

<Empfaenger> ::=              "<receiver>"
                                                    (<Agentenidentifikator>)
                              "</receiver>"

<Agentenidentifikator> ::=    "<agent-identifier>"
                                                    (<Agentenname>)
                              "</agent-identifier>"

<Agentenname> ::=             "<name id=" <String> ">"

<Protokoll> ::=               {Protokolle}

<Sprache> ::=                 {Sprachen}

<Ontologie> ::=               {Ontologien}

<Dienstbeschreibung> ::=      "<service-description>"
                                                    (<Diensttyp>)
                              "</service-description>"

<Diensttyp> ::=               "<type id=" <String> ">"

<String> ::=                  Eine Zeichenkette in Anfuehrungszeichen, die nur Buchstaben,
                              Zahlen und Leerzeichen enthaelt.
```

Abbildung 6.7-8: ***Darstellung der Sprache zur Beschreibung von Prozessen (PML)***
Quelle: Eigene Darstellung, in Anlehnung an *Meyer* (2004, 54) und *Bergmann/Klinger/Winkler* (2006, 16-21)

6.7.6.2 Architektur

Bei Workflow-Management-Systemen werden hinsichtlich ihrer Anwendung meist die beiden Phasen Modellierung und Ausführung (Leymann/Altenhuber 1994; Jablonski 1997, 74) von Prozessen differenziert. Analog dazu wird die in Abschnitt 6.7.6.1 beschriebene Sprache zur Prozessmodellierung in diesem Abschnitt durch eine Architektur zur Transformation und Persistenzsicherung der Prozessmodelle sowie für die Prozesssteuerung ergänzt. Grundlagen für diese Architektur sind Erkenntnisse der Prozesssteuerung und Workflow Engines (Curtis/Keller/Over 1992; Jablonski/Böhm/Schulze 1997; Leymann/Roller 1999), die entsprechend den Anforderungen in einem MAS angepasst sind (Krcmar/Horn 2003, 24).

Anforderungen an die Architektur zur Prozesssteuerung werden durch die folgenden Funktionalitäten bestimmt:

- Um die in der PML definierten Prozesse durch eine Verarbeitungseinheit ausführen zu lassen, wird bei *Meyer* (2004, 23-38) eine Implementierung beschrieben. Diese ist Grundlage für eine Erweiterung in der vorliegenden Arbeit.
- Ausgangspunkt für die automatisierte Prozessunterstützung ist ein mit dem ARIS Toolset modellierter Prozess in EPK-Notation.
- Das Design ist so zu gestalten, dass auch Prozessunterbrechungen und ihre Wiederaufnahme (Bergmann/Klinger/Winkler 2006, 16) unterstützt werden, um die Anforderungen aus der Domäne adäquat abbilden zu können.
- Die Prozessinformationen sollen in einer zentralen Datenbasis persistent abgelegt werden (Bergmann/Klinger/Winkler 2006, 14), um für alle Agenten als Informationsbasis zu fungieren. Die Entscheidung für eine zentrale Datenbank schränkt die Qualität der Ergebnisse der vorliegenden Arbeit nicht ein, weil sich eine verteilte Datenbank aus der Perspektive der Applikation äquivalent wie eine zentrale Datenbank verhält und damit Transparenz für den Zugriff auf die Datenbank gewährleistet. Dementsprechend wird mit dieser Design-Entscheidung lediglich die Implementierungskomplexität reduziert.
- Visualisierung der Prozessinformationen und des aktuellen Verarbeitungszustandes an der Benutzeroberfläche

Um diese Anforderungen zu erfüllen, werden die folgenden Anpassungen bei PML vorgenommen (Bergmann/Klinger/Winkler 2006, 17-18):

- Ergänzung der in Abschnitt 6.7.6.1 beschriebenen PML-Beschreibung mit dem Konnektor XOR aus der EPK-Modellierung
- Gestaltung des constraints-Elements nach dem Composite-Entwurfsmuster (Gamma et al. 1995, 163-173): Diese Anpassung ist erforderlich, weil komplexe Prozesse komposite constraints erfordern.

- Für eine geeignete Visualisierung der Prozesse wird im process-Element das Attribut epkNodeId gespeichert, welches mit einer EPK-Funktion in der AML-Beschreibung[67] assoziiert ist.
- Im Unterschied zu der in Abbildung 6.7-8 beschriebenen PML-Beschreibung wird in der Umsetzung auf die Unterstützung des Attributs type im Element agent verzichtet (Bergmann/Klinger/Winkler 2006, 18). Für eine Erläuterung dieser Implementierungsentscheidung siehe Abschnitt 6.8.7.

Die aus diesen Anforderungen abgeleitete Architektur der Prozesssteuerung ist in Abbildung 6.7-9 dargestellt und wird im Folgenden beschrieben: Die AML-Beschreibung wird aus der visuellen Modellierung mit dem ARIS Toolset in der Form einer Datei generiert. Damit kann vom Domänenexperten von der Implementierung abstrahiert und ausschließlich das Domänenproblem fokussiert werden. Diese AML-Beschreibung wird in eine EPK-Objektrepräsentation transformiert. Um zu vermeiden, dass fehlerhafte AML-Dateien in die Prozesssteuerung eingelesen werden und damit zur Laufzeit Fehlerpotenziale entstehen würden, werden zur Validierung geeignete Schema-Dateien verwendet. Grundlage für diese Validierung sind die bei *Nüttgens/Rump* (2002, 67-71) beschriebenen und in Tabelle A.9-2 zusammengefassten Regeln.

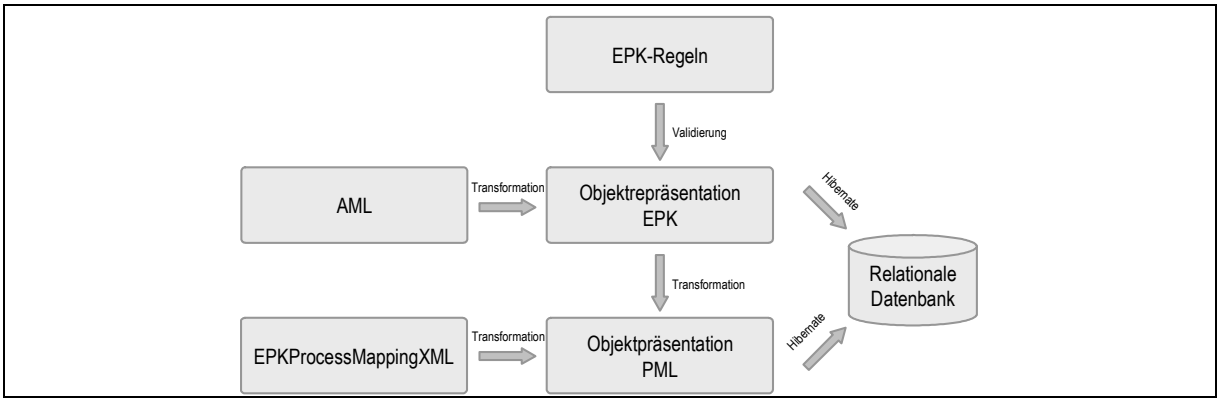

Abbildung 6.7-9: ***Transformation der Prozessinformation***
Quelle: Eigene Darstellung

Um die weiteren Komponenten der Architektur in Abbildung 6.7-9 beschreiben zu können, werden im Folgenden zunächst die dafür erforderlichen Grundlagen dargestellt: Die AML-Beschreibung eines klinischen Pfades ist nicht geeignet, um diese unmittelbar in eine entsprechende PML-Beschreibung zu transformieren (Bergmann/Klinger/Winkler 2006, 14). Dazu wäre die Ableitung von Aktionen erforderlich, die für eine EPK-Funktion von einem Agenten in einem geeigneten Behaviour auszuführen wären (Bergmann/Klinger/Winkler 2006, 14). Um eine automatisierte Transformation von AML zur geforderten Eingabe für die Prozesssteuerung in PML zu konstruieren, wird von *Bergmann/Klinger/Winkler* die Beschreibungssprache EPKProcessMapping mit den folgenden Eigenschaften definiert (2006, 14 und 21-22):

[67] Ereignisgesteuerte Prozessketten, die mit dem Werkzeug ARIS-Toolset modelliert werden, können über eine XML-Datei exportiert werden. Diese XML-Datei liegt im Format AML (ARIS Markup Language) vor. Für eine Beschreibung von AML wird auf *Mendling/Nüttgens* (2004) verwiesen.

- Bestimmung der Behaviours und Agenten für eine EPK-Funktion (Bergmann/Klinger/Winkler 2006)
- Anlehnung des XML-Formats für EPKProcessMapping (Bergmann/Klinger/Winkler 2006, 21-24) an PML (Meyer 2004, 18-22)
- Assoziation einer EPK-Funktion jeweils mit einem EPKProcessMapping-Element in der EPKProcessMapping-Datei
- Erweiterung um das Element EPKEventConstraint-Element, um geeignete Vorbedingungen bei XOR-Verzweigungen oder -Zusammenführungen modellieren zu können. Damit wird ein deterministisches Verhalten erreicht. Dazu werden Constraints, die für eine EPK-Funktion erfüllt sein müssen, bevor diese ausgeführt werden kann, auch für Ereignisse ergänzt. In PML sind diese Constraints nicht für Ereignisse definiert.

Aus diesen Anforderungen wird die in Abbildung 6.7-10 dargestellte Struktur der Elemente in einer EPKProcessMappingXML-Datei abgeleitet. Die folgende Beschreibung dieser Struktur ist zusammengefasst aus der Darstellung bei *Bergmann/Klinger/Winkler* (2006, 21-22):

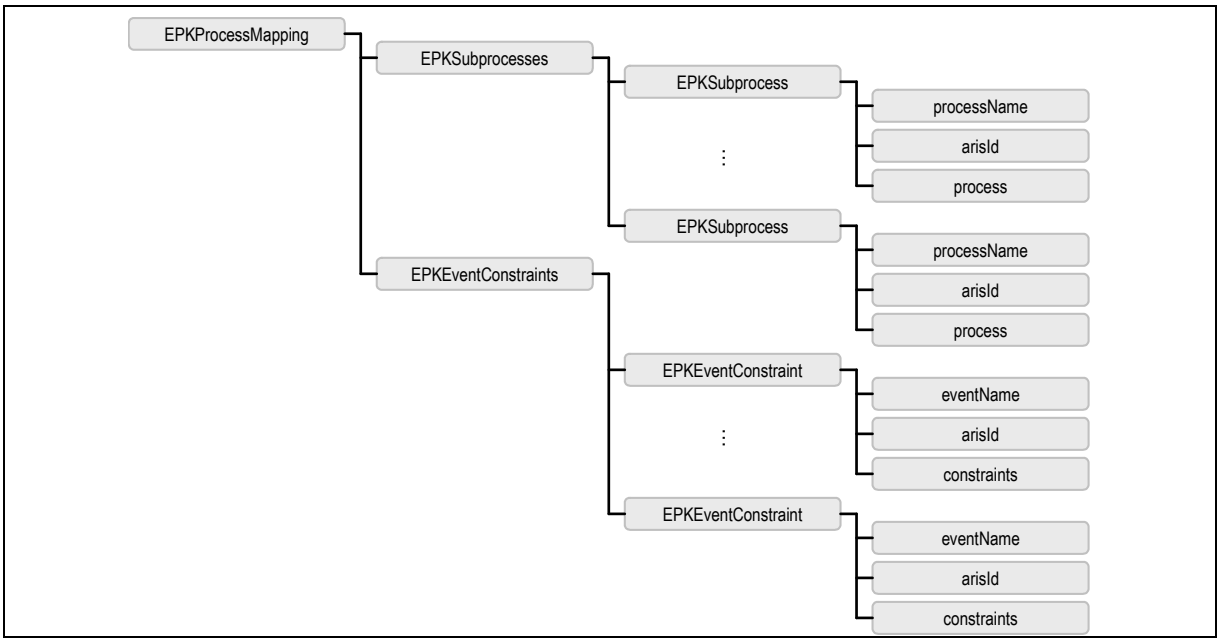

Abbildung 6.7-10: ***Struktur des EPKProcessMappingXML-Formats***
Quelle: *Bergmann/Klinger/Winkler* (2006, 21)

In einem EPKSubprocess-Element wird die Relation zwischen einer EPK-Funktion und einem process-Element im PML-Format und damit einem Agenten mit seinen Behaviours beschrieben. Dazu enthält dieses Element die Attribute processName und arisId, über die eine EPK-Funktion bestimmt werden kann. Weiterhin wird durch ein process-Element der entsprechende Agent spezifiziert, der darüber mit der relevanten EPK-Funktion assoziiert ist. In dem EPKEventConstraint-Element werden die Vorbedingungen (Element constraints) für EPK-Ereignisse angegeben, die nach einer XOR-Verzweigung folgen bzw. vor einer XOR-Zusammenführung auftreten. Über die Attribute eventName bzw. arisId werden EPK-Ereignisse referenziert.

Mit der beschriebenen EPKProcessMapping-Erweiterung kann nun die in Abbildung 6.7-9 dargestellte Architektur abschließend beschrieben werden: Mit der EPKProcessMapping-Beschreibung kann aus der Objektrepräsentation der EPK eine PML-Repräsentation generiert werden, die wiederum von der Prozesssteuerung (Meyer 2004) verarbeitbar ist und die Information für die Abarbeitung des gewählten Behandlungsprozesses darstellt (Bergmann/Klinger/Winkler 2006, 14). Darin sind die durch die modellierten EPKn bestimmten Bedingungen entsprechend den Ausdrucksmöglichkeiten der PML formuliert: Werden in ARIS z.B. konsekutive Funktionen modelliert, erfolgt die Abbildung in PML derart, dass die Vorbedingung der nachgelagerten Funktion erst dann erfüllt ist, wenn die vorhergehende Funktion abgeschlossen ist.

Für eine automatisierte Ausführung eines Prozesses würden die PML-Objektrepräsentation und ihre Speicherung in einem zentralen Repository bzw. einer Datenbank sowie die aus diesen Daten generierte Datei als Eingabe für die Prozesssteuerung ausreichen. Weil bei der Transformation von AML bzw. Objektrepräsentation der EPK in die Objektrepräsentation der PML die Informationen über die EPK-Visualisierung nicht erhalten bleiben (Bergmann/Klinger/Winkler 2006, 14), werden beide Objektstrukturen über Hibernate (Red Hat Middleware LLC 2006) zur Persistenzsicherung in einer relationalen Datenbank abgelegt. Weiterhin kann damit eine geeignete grafische Visualisierung des aktuell verarbeiteten Prozesses umgesetzt werden. Insgesamt wird das Design somit derart gestaltet, dass die Prozessinformation mehrfach wieder verwendet werden kann, ohne die beschriebene Transformation wiederholt ausführen zu müssen (Bergmann/Klinger/Winkler 2006, 14). Auch um Prozesse an einer gespeicherten Stelle fortführen zu können, ist in der Datenbank die Speicherung des aktuellen Prozesszustandes erforderlich.

Um die Verarbeitung von persistenten Objekten zu vereinfachen, wird das objektrelationale Mapping von Hibernate verwendet. In Anlehnung an *Bergmann/Klinger/Winkler* können mit dieser Vorgehensweise die folgenden Vorteile erzielt werden (2006, 26-27):

- Daten müssen nicht über Datenbankabfragen aus der Datenbank gelesen oder in die Datenbank geschrieben werden. Mit der Hibernate-Schicht zwischen einer relationalen Datenbank und der eigentlichen Applikation können Objekte verarbeitet werden, ohne ihre Daten zuvor manuell aus der Datenbank zu lesen oder in diese zu schreiben.
- Über Hibernate werden die Tabellendefinitionen für die Datenbank aus dem Programmtext und einer entsprechenden Abbildungsvorschrift generiert. Diese Abbildungsvorschrift wird in den Programmtext in Form von Annotationen eingefügt. Der Implementierungsaufwand für die Konstruktion geeigneter Tabellen in der Datenbank wird damit reduziert.
- Der Einsatz der Hibernate-Zwischenschicht gewährleistet neben der Abstraktion für den Anwendungsentwickler die Unabhängigkeit von unterschiedlichen SQL-Derivaten sowie eine angemessene Architektur für die Anwendung über die Aufteilung in Persistenz- und andere Schichten.
- Mit dem Einsatz der Hibernate-Schicht wird die Anwendung vom Einsatz bestimmter relationaler Datenbanken entkoppelt. Damit wird die Abhängigkeit von Produktweiterentwicklungen reduziert.

- Weiterhin sind bei Datenbanken bereits etablierte Mechanismen wie z.B. Transaktionen (siehe dazu bspw. Weikum/Vossen 2002) implementiert, die mit Hibernate in die entwickelte Anwendung integriert werden können. Somit wird der Implementierungsaufwand reduziert.

Um für die Prozessunterstützung die entsprechende Information aus der Datenbank zur Laufzeit der Applikation zu erhalten (Phase der Prozessausführung, siehe Abbildung 6.7-11), werden die folgenden Schritte (siehe dazu auch Bergmann/Klinger/Winkler 2006, 14) ausgeführt:

- Die in der Datenbank gespeicherte Information wird in eine Datei im PML-Format transformiert, welche als Eingabe für die Prozesssteuerung als Bestandteil des AMD fungiert.
- Abhängig von der Ereignisgenerierung durch Benutzer über die Benutzerschnittstelle oder andere Systeme (hier simuliert durch SeSAm, siehe dazu auch Abschnitt 6.8.8) wird der gewählte Prozess gesteuert.
- Die EPK-Objektrepräsentation dient zur Visualisierung des Prozesses, um für den Anwender an der Benutzeroberfläche den aktuellen Status in der Behandlung anzuzeigen.

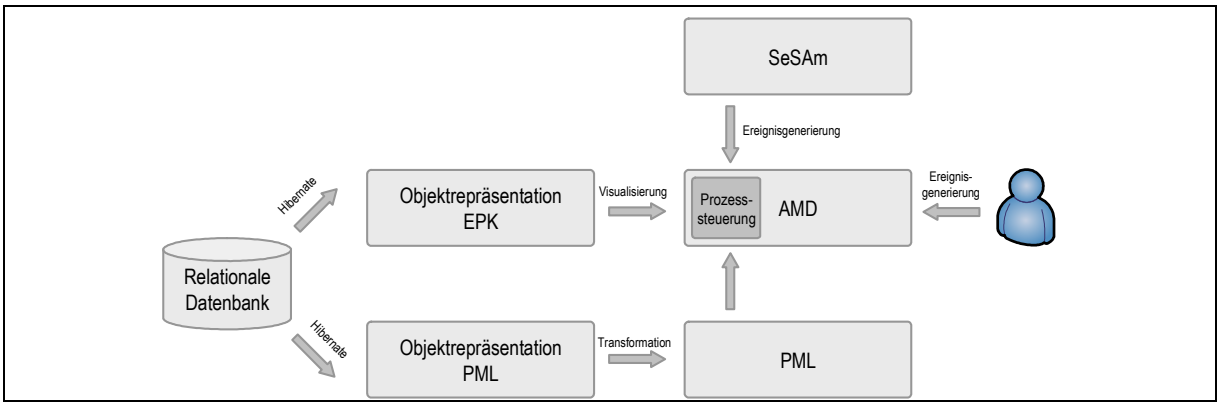

Abbildung 6.7-11: ***Nutzung der Prozessinformation zur Prozesssteuerung***
Quelle: Eigene Darstellung

Die Darstellung der Architektur der Prozesssteuerung und die durch sie zur Verfügung gestellten Funktionalitäten zeigen, dass eine Trennung der Domänenmodellierung und der Definition bzw. Implementierung der Prozesssteuerung vorgenommen werden kann. Dazu wurde insbesondere eine geeignete Transformationsvorschrift beschrieben, die die in einem Werkzeug modellierten ARIS-Prozesse im AML-Format in ein durch die Prozesssteuerung verarbeitbares Datenmodell abbildet. Diese Daten werden neben der Objektrepräsentation der modellierten EPK in einer Datenbank abgelegt. Zur Laufzeit werden diese Daten ausgelesen, um den gewählten Prozess steuern und visualisieren zu können.

Weiterhin wurden bei Architekturentscheidungen Standards und Erkenntnisse der Prozesssteuerung sowie Workflow Engines berücksichtigt und auf Agentensysteme angepasst (Krcmar/Horn 2003, 23-24). Somit konnte demonstriert werden, dass auch agentenbasierte Anwendungen von den genannten Ergebnissen profitieren können.

6.7.7 Zusammenfassung zur Designbeschreibung

In den Abschnitten 6.7.1 mit 6.7.6 wurde das Design von Applikationen beschrieben, mit denen alle Anforderungen des Kriterienkatalogs aus Abschnitt 6.3.6 erfüllt werden. Damit wird die Konkretisierung des Anforderungskatalogs für ein Informationssystem (siehe dazu Abschnitt 6.3.6) aus Design-Perspektive umgesetzt. Somit ist die in Abschnitt 1.4 genannte

Forschungsfrage 2 beantwortet: Im Zentrum der vorgestellten Lösungsansätze steht das AMD auf der Basis einer Java-Laufzeitumgebung (siehe Abschnitt 6.3.1, Anforderung 12 aus Abschnitt 6.4.2), an das die beschriebenen Anwendungen angekoppelt werden können. Damit konnte demonstriert werden, dass ein AMD aus Design-Perspektive als ein zentrales Element in einer künftigen Ausprägung einer Informationssystemlandschaft im Gesundheitswesen fungieren kann. Das AMD kann deshalb als ein adäquates Lösungskonzept zur Beseitigung identifizierter Defizite im Gesundheitswesen betrachtet werden. Im Zusammenhang mit den Ausführungen zur EAI bzw. SOA in den Abschnitten 3.2.4.1 bzw. 3.2.4.2 können Elemente dieser Ansätze in der vorliegenden Arbeit durch die Prozess- und Datenintegration sowie die Kapselung von Funktionalität entsprechend des SOA-Dienstkonzepts und eine Aufbereitung von Daten in einem Portal gemäß der SOA Desktop Integration umgesetzt werden.

Auf der Grundlage des in Abschnitt 6.7 beschriebenen Designs werden im folgenden Abschnitt 6.8 die jeweiligen Implementierungsdetails der Applikationen dargestellt, um die Grundlage für die Beantwortung der Forschungsfrage 3 zu legen.

6.8 Beschreibung der Implementierung

In den folgenden Abschnitten 6.8.1 mit 6.8.7 werden jeweils die Implementierungen der Applikationen auf Basis der JADE-Plattform (Anforderung 19 aus Abschnitt 6.4.2) und Java-Laufzeitumgebung (Anforderung 12 aus Abschnitt 6.4.2) beschrieben[68]. Damit werden für die Agentenanwendungen insbesondere die Interaktionen dargestellt, die entsprechend der ArBaCon-Methode in Etappe 2 zu spezifizieren sind.

6.8.1 Desktop-Anwendung zur Informationsaggregation

In diesem Abschnitt werden die Verarbeitungsschritte der Implementierung beschrieben, die zur Realisierung des folgenden Szenarios aus der OP-Vorbereitung (Abschnitt 6.6.1) des gewählten Anwendungsfalles erforderlich sind: Ein Anaesthesist in einem Krankenhaus möchte zur Entscheidungsfindung, ob die geplante Operation stattfinden kann, die bisherige Krankheitsgeschichte des Patienten in Erfahrung bringen. Dazu erfolgt die Anmeldung des Anwenders am Informationssystem. Weiterhin sind der HBA sowie die eGK in den simulierten Kartenleser einzulegen. Diese Karten sind jeweils durch eine PIN-Eingabe zu autorisieren. In einem entsprechenden Menüeintrag wählt der Anwender nach der Öffnung der Patientenakte die Aggregation der Patientendaten aus der bisherigen Krankheitsgeschichte aus. Dazu muss über den Konnektor des Krankenhauses zunächst die Verweisliste aus der eGK ausgelesen werden. Sobald diese Daten vorliegen, können die Anfragen an das lokale KIS direkt sowie indirekt an das entfernte PVS über den lokalen und entfernten Konnektor der TI gestellt werden. Die aus den Informationssystemen ausgelesenen Daten werden in das CDA-Format transformiert und mit LOINC-Codes versehen. Die Ergebnisse der Suche werden direkt bzw. über den entfernten und lokalen Konnektor wieder zurück an die Anwendung zur Anzeige der virtuellen Patientenakte gesendet. Dort werden diese Daten empfangen und zur Visualisierung aufbereitet.

[68] Zur Verbesserung der Verständlichkeit der Ausführungen zur Implementierung wird empfohlen, parallel zur vorliegenden Arbeit den zugehörigen Programmtext zu betrachten.

Entsprechend der in Abschnitt 6.6.6 dargestellten Verteilung der Komponenten werden zu Beginn der Anwendung auf den jeweiligen Agentenplattformen mehrere Agenten gestartet. Die Plattformen und die auf diesen befindlichen Agenten sind in Tabelle 6.8-1 dargestellt. Für die Plattform Telematik wird kein eigener Agent gestartet. Die Begründung dafür ist in Abschnitt 6.6.6 beschrieben. Für die genannten Umgebungen wird jeweils eine eigene Agentenplattform gestartet, um auch eine lokale Verteilung der einzelnen Systeme unterstützen zu können.

Plattform	Registrierte Agenten
Hospital	PersonalAssistantGuiAgent
	HConnectorAgent
	HISWrapper
	GatewayAgent
	CardReaderGuiAgent
	ResourceSchedulingTaskAgentSurgery
	ResourceSchedulingTaskAgentECG
	ResourceSchedulingTaskAgentX-raying
	ResourceSchedulingTaskAgentCT
General Practitioner	GPISWrapper
	GPConnectorAgent

Tabelle 6.8-1: ***Überblick über Agenten, die bei der Initialisierung der Anwendung gestartet werden***
Quelle: Eigene Darstellung

Die Beschreibung der Funktionalitäten der Konnektoren HConnectorAgent und GPConnectorAgent sowie ihrer Behaviours ist ausführlich in Abschnitt 6.8.3 dargestellt. Deshalb wird im Folgenden auf eine entsprechende Ausführung verzichtet. Ebenso wird für die Beschreibung der Funktionalitäten der Agenten ResourceSchedulingTaskAgent<Modality> und GatewayAgent auf die Abschnitte 6.8.5 bzw. 6.8.6 verwiesen.

Im Folgenden wird die Initialisierung des PersonalAssistantGuiAgent beschrieben (für eine Beschreibung der Verarbeitungsschritte siehe auch Anhang A.5.9). In dessen setup-Methode wird in der Instanz der zugehörigen Benutzeroberfläche JPersonalAssistantGui die Methode initAgent aufgerufen. Mit dem Aufruf dieser Methode wird der Agent CardReaderGuiAgent[69] (Simulation des Kartenlesegeräts, für eine Darstellung der Verarbeitungsvorschrift siehe Anhang A.5.8) generiert. Weiterhin wird in der setup-Methode der Klasse PersonalAssistantGuiAgent die Benutzeroberfläche JPersonalAssistantGui des PersonalAssistantGuiAgents konstruiert. In der init-

[69] Die Funktionalität des CardReaderGuiAgents ist ausführlich in Abschnitt 6.8.3 beschrieben bzw. in Anhang A.5.8 zusammengefasst.

Methode der Klasse JPersonalAssistantGui werden dazu die grafischen Elemente der Benutzeroberfläche zusammengestellt und diese schließlich zur Anzeige gebracht (siehe Abbildung 6.8-1). Außerdem wird in der setup-Methode das Behaviour ReceiveDeliverDocuments für den Agenten PersonalAssistantGuiAgent aktiviert.

Abbildung 6.8-1: ***Screenshot[70] der Desktop-Anwendung vor der Anmeldung des Anwenders***
Quelle: Eigene Darstellung

Sobald im Programmmenü vom Anwender der Eintrag Anmelden gewählt wird, wird über den assoziierten Listener ein neues Objekt vom Typ JPersonalAssistantGuiLoginPanel erzeugt. In dessen Konstruktor wird die Methode init aufgerufen. Darin werden die grafischen Elemente der Benutzeroberfläche zur Anmeldung an der Applikation zusammengestellt und diese schließlich visualisiert. Sobald die Login-Daten (Benutzername und Passwort) eingegeben sind und auf den OK-Button geklickt wird, wird über dessen Listener die Methode checkLogin aufgerufen. In dieser Methode werden die Login-Daten aus dem Dialog mit den in einer XML-Datei hinterlegten Informationen überprüft. Stimmen die Daten überein, erfolgen die Anmeldung und damit eine entsprechende Aktivierung der Benutzeroberfläche der Hauptanwendung. Bei der beschriebenen Überprüfung der Berechtigung werden keine agentenspezifischen Funktionalitäten aufgerufen, weil diese Überprüfung lediglich die lokale Anwendung betrifft und damit der Aufwand für die Implementierung gegenüber einer agentenbasierten Umsetzung reduziert wird.

Nach der Anmeldung an der Applikation können die Gesundheitskarten in den (simulierten) Kartenleser eingelegt werden. Dazu erscheint beim Klick auf den entsprechenden Menüeintrag ein Dialog, der die Benutzeroberfläche für den simulierten Kartenleser darstellt. Dafür wird über den zugehörigen Listener die Methode startNewGUI aus der Klasse CardReaderGuiAgent aufgerufen. In dieser Methode wird eine neue Instanz des Kartenlesers (JPersonal-

[70] Das auf diesem Screenshot dargestellte Logo wurde freundlicherweise von Herrn Matthias Baume vom Lehrstuhl für Wirtschaftsinformatik an der Technischen Universität München erstellt.

AssistantGuiCardReader) generiert. In dem Konstruktor der Klasse JPersonalAssistantGuiCardReader wird die Methode createGui aufgerufen, in welcher die grafischen Elemente für die Benutzeroberfläche generiert werden. In der Methode startNewGui wird nach der Konstruktion der GUI für den Kartenleser die Benutzeroberfläche angezeigt.

Sobald die Patientenkarte eingelegt wird, wird über den zugehörigen Listener (actionPerformed) die Methode patientPinCheck aufgerufen. In dieser Methode wird die eingegebene PIN mit derjenigen in einer lokal abgelegten XML-Datei verglichen. Erst nach erfolgreicher Eingabe der richtigen PIN kann die Funktionalität der eGK aktiviert werden. Bei der Benutzerkarte, d.h. dem Heilberufsausweis, wird über den assoziierten Listener die Methode userPinCheck aufgerufen. Dabei wird analog zur Bearbeitung in der Methode patientPinCheck verfahren. Dabei ist zu beachten, dass die Benutzerkarte durch das System entsprechend der Anmeldung bei der Anwendung selektiert wird.

Erst wenn beide PINn korrekt eingegeben werden, wird der Weiter-Button der Benutzeroberfläche für den Kartenleser aktiviert. Über den zugehörigen Listener dieses Buttons werden in dem Ontologieelement CardReaderStatus der Klasse CardReaderGuiAgent die folgenden Ontologieeinträge vorgenommen:

- Patientenidentifikation
- Patientenname
- Benutzername
- Benutzerrolle

Nach dem erfolgreichen Einlegen der Patienten- und Nutzerkarten ist die Öffnung der Patientenakte bzw. die Konstruktion des AMDs über den entsprechenden Menüeintrag möglich. Der mit diesem Eintrag assoziierte Listener ruft die Methode startAMD des PersonalAssistantGuiAgent auf. Darin wird über den Methodenaufruf startAMDInstance die Initialisierung des Agenten ActiveMedicalDocument durchgeführt. In dessen setup-Methode wird der folgende Dienst dieses Agenten in der Umgebung Hospital registriert:

- Typ Open Electronic Healthcare Record
- Name EHR AMD Service

Nach der Registrierung erfolgt die Aktivierung der Behaviours ReceiveKillAMD und RecieveOpenEHR (für eine Darstellung der Verarbeitungsvorschrift siehe Anhang A.5.12). Im Behaviour ReceiveKillAMD werden Nachrichten mit der folgenden Spezifikation empfangen:

- ConversationID[71] killAMD
- Performativ[72] REQUEST

Sobald eine Nachricht von diesem Typ eintrifft, werden die internen Agenten des AMD sowie das AMD selbst beendet.

---

[71] Mit der ConversationID wird die Kennung einer Nachricht beschrieben (Hillebrand 2006, 78).

[72] Performative definieren den Typ einer Nachricht (Hillebrand 2006, 78). Für weitere Ausführungen zu Performativen siehe auch Abschnitt 2.3.6.

Im Behaviour RecieveOpenEHR wird auf ACL-Nachrichten mit der folgenden Spezifikation gewartet:

- ConversationID openEHR
- Performativ REQUEST

Sobald eine entsprechende Nachricht empfangen wird, wird die in dieser Nachricht enthaltene Patientenidentifikation lokal gespeichert und so das AMD für den aktuellen Patienten personalisiert. Es folgt der Aufruf der Methode startInternalAgents, durch die die folgenden Agenten des AMD gestartet und mit der aktuellen Patientenidentifikation belegt werden:

- DataWrapper (Anforderung 5 aus Abschnitt 6.4.1, für eine Beschreibung der Verarbeitungsschritte siehe Abschnitt A.5.2)
- DataRetrievalManager (Anforderung 5 aus Abschnitt 6.4.1, siehe auch Abschnitt A.5.1)
- ViewManager (Anforderung 5 aus Abschnitt 6.4.1, siehe auch Abschnitt A.5.5)
- AppointmentMonitor (Anforderung 5 aus Abschnitt 6.4.1)
- AppointmentManager (Anforderung 5 aus Abschnitt 6.4.1)
- Subscriber (Anforderung 5 aus Abschnitt 6.4.1, siehe Abschnitt 6.8.6)
- ProcessMasterAgent (Anforderung 5 aus Abschnitt 6.4.1, siehe Abschnitt 6.8.7)

In den setup-Methoden dieser Agenten werden die in Tabelle 6.8-2 dargestellten Initialisierungen durchgeführt. Die Funktionalitäten von AppointmentManager, AppointmentMonitor, Subscriber sowie ProcessMasterAgent werden ausführlich in den Abschnitten 6.8.5.1, 6.8.6 bzw. 6.8.7 beschrieben.

Agent	Registrierter Dienst	Initial aktivierte Behaviours
AppointmentManager	• Typ EHR<Patientenidentifikation> • Name ManageAppointments • Registriert in der Umgebung Hospital	• ListenForSesamAppointmentRequest • ListenForOPAppointmentRequest • ReceiveQueryPatientSchedule • ReceiveRescheduleRequest • ReceiveCreateNewAppointment • ReceiveChangeAppointment • ReceiveDeleteAppointment
AppointmentMonitor	• Typ EHR <Patientenidentifikation> • Name GetAppointment • Name NewAppointment • Jeweils registriert in der Umgebung Hospital	ReceiveUpdateSchedule
DataRetrievalManager	• Type EHR<Patientenidentifikation> • QueryPatientData • Registriert in der Umgebung Hospital	ReceiveQueryPatientData
DataWrapper	• Keine Registrierung, da keine öffentlichen Dienste angeboten werden • Determinierung der Agentennamen, die Anfragen an den DataWrapper senden dürfen: AMD_DataWrapper, AMD_DataRetrievalManager, AMD_ViewManager, AMD_AppointmentManager	• ReceiveLocateResources • ReceiveRequestLinks • ReceiveCheckAccess • ReceiveAddDocuments

Subscriber	• Typ web-service • Name OPERATION1 (Initialisierung der Dokumentenaggregation und Aktivierung des Behaviours ReceiveDeliverDocuments für das Empfangen der Dokumente der virtuellen Patientenakte) • Jeweils registriert in der Umgebung Hospital	CyclicBehaviour (Dispatching der vom WSIG Gateway erhaltenen Anforderungen an die Agentenanwendung)
ProcessMasterAgent	• Type Electronic Healthcare Process Execution • Name EHR Process Service • Registriert in der Umgebung Hospital	Keine initial aktivierte Behaviours

Tabelle 6.8-2: ***Initialisierungen und Behaviours der Management-Agenten***
Quelle: Eigene Darstellung

Nach diesen Initialisierungen wird in der setup-Methode der Oberklasse AMDManagementAgent dieser Agenten bzw. direkt in der Klasse Subscriber die folgende ACL-Nachricht konstruiert:

- Performativ INFORM
- Content StartInternalServiceAgent

Diese Nachrichten werden von den initialisierten internen Agenten an den Agenten ActiveMedicalDocument gesendet und dort im Behaviour WaitOnNotifications entsprechend bearbeitet. Konnten alle internen Agenten durch den Aufruf der Methode startInternalAgents erfolgreich gestartet werden, wird das Behaviour WaitOnNotifications aktiviert. In diesem Behaviour werden Nachrichten mit folgendem Typ erwartet:

- Performativ INFORM
- Content StartInternalServiceAgent

Sobald alle internen Agenten des AMD gestartet sind, wird vom AMD an den Agenten, der eine Nachricht an das Behaviour RecieveOpenEHR sendete, eine entsprechende Nachricht mit der folgenden Spezifikation gesendet, womit die Initialisierung des AMD beendet ist:

- Performativ INFORM
- inReplyTo startEHR<aktuelle Patientenidentifikation>

Weiterhin wird in der Methode startAMD des PersonalAssistantGuiAgent das Behaviour SendStartAMD aktiviert. Darin werden schließlich die Behaviours ReceiveAMDReady sowie SendOpenEHR aktiviert. Abschließend werden über den Listener des Menüeintrags zum Öffnen der Patientenakte weitere Menü- und Navigationsleisteneinträge aktiviert.

Im Behaviour ReceiveAMDReady wird vom PersonalAssistantGuiAgent auf die folgende Nachricht gewartet, d.h. so lange, bis das entsprechende AMD geöffnet ist bzw. seine internen Agenten initialisiert sind:

- Performativ INFORM
- inReplyTo startEHR<aktuelle Patientenidentifikation>

Im Behaviour SendOpenEHR (aktiviert im Behaviour SendStartAMD, siehe weiter oben) wird in der setup-Methode die folgende ACL-Nachricht konstruiert:

- Performativ REQUEST
- ConversationID openEHR

Diese Nachricht wird vom Agenten PersonalAssistantGuiAgent an den Agenten ActiveMedical-Document gesendet und dort entsprechend verarbeitet. Im Behaviour ReceiveOpenEHR des ActiveMedicalDocuments erfolgt diese Verarbeitung wie folgt (für Detailausführungen dazu siehe weiter oben):

- Starten der internen Agenten
- Warten, bis diese Agenten gestartet sind
- Senden der Nachricht INFORM mit der ConversationID startEHR<aktuelle Patientenidentifikation> an den PersonalAssistantGuiAgent

Wird auf den Menüeintrag „Daten aus früheren Episoden aggregieren“ geklickt, wird ein neues Objekt vom Typ JPersonalAssistantGuiPatientID generiert und in dieser Klasse die Methode setNewPatientId aufgerufen (siehe Abbildung 6.8-2). In dieser Methode wird zunächst die Methode lookUpEHR im Agenten PersonalAssistantGuiAgent aufgerufen. Darin wird nach dem Dienst mit dem Namen QueryPatientData und dem Typ EHR<aktuelle Patientenidentifikation> gesucht. Dieser Dienst wird vom Management-Agenten DataRetrievalManager angeboten. Letzterer repräsentiert den Einstiegspunkt zum ActiveMedicalDocument des Patienten (Anforderung 15 aus Abschnitt 6.4.2). Verläuft diese Anfrage erfolgreich, wird als Ergebnis der Methode lookUpEHR der Wert OK zurückgeliefert. Sobald dieser Rückgabewert vorliegt, wird in der Methode setNewPatientId die Methode sendQueryPatientData in der Klasse PersonalAssistant-GuiAgent aufgerufen. In dieser Methode wird lediglich das Behaviour SendQueryPatientData aktiviert.

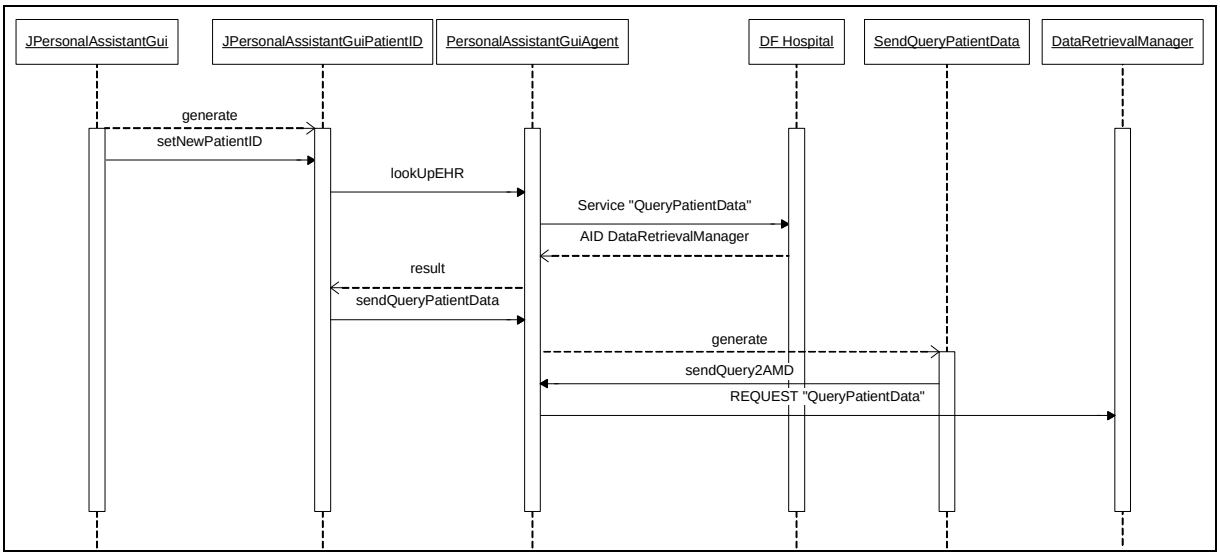

Abbildung 6.8-2: Interaktionen zum Öffnen und Suchen der Patientenakte
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

In diesem Behaviour verzweigt der Kontrollfluss zur Methode sendQuery2AMD in der Klasse PersonalAssistantGuiAgent. In dieser Methode wird zunächst ein Ontologieobjekt vom Typ QueryPatientData generiert. Dieses wird nun mit den gesuchten Typen von Patientendokumenten belegt. Weiterhin werden die Patientenidentifikation, der Benutzername und die Benutzerrolle entsprechend den aktuell eingelegten Karten bzw. dem Login des Benutzers belegt. Es folgt eine Konstruktion einer ACL-Nachricht mit der folgenden Spezifikation:

- Performativ REQUEST
- ConversationID QueryPatientData
- Ontologieobjekt QueryPatientData

Diese Nachricht wird an den durch die Methode lookUpEHR determinierten DataRetrievalManager gesendet und dort weiterverarbeitet (Beschreibung dazu weiter unten).

Da das AMD nun nach dem Starten der internen Agenten verarbeitungsbereit ist, wird im Folgenden die eigentliche Anfrage zur Aggregation von Patientendokumenten beschriebenen. Diese Schritte sind abstrakt in Abbildung 6.8-3 dargestellt.

Die vom PersonalAssistantGuiAgent an den DataRetrievalManager gesendete Nachricht mit dem Performativ REQUEST und der ConversationID QueryPatientData (Konstruktion der Nachricht siehe weiter oben) wird dort wie folgt verarbeitet: Im Behaviour ReceiveQueryPatientData wird für jeden REQUEST eine eindeutige Identifikation generiert, um die weitere Bearbeitung der Nachrichten zu ermöglichen. Dazu werden lokal die folgenden Informationen für jeden eingehenden REQUEST gespeichert:

- Typen der Requests nach den gesuchten Dokumenten für jeden anfragenden Agenten
- Adresse für jeden anfragenden Agenten

Abschließend wird im Behaviour ReceiveQueryPatientData das Behaviour SendLocateResources aktiviert. In diesem Behaviour wird die folgende ACL-Nachricht konstruiert:

- Performativ REQUEST
- ConversationID LocateResources

Diese Nachricht wird vom DataRetrievalManager an den DataWrapper gesendet und dort weiterverarbeitet. Es folgt im Behaviour SendLocateResources die Aktivierung des Behaviours ReceiveLocateResources.

Im Folgenden wird die Bearbeitung der Nachricht REQUEST und LocateResources im DataWrapper beschrieben: Diese Nachricht wird im Behaviour ReceiveLocateResources verarbeitet. Zunächst wird überprüft, ob der sendende Agent in der Liste der erlaubten Sender ist. Dazu wird in der Methode checkAccess überprüft, ob der anfragende Agent zu den internen Agenten zählt, wie sie in Tabelle 6.8-2 beschrieben sind. Da im vorliegenden Fall der DataRetrievalManager diesen Agenten zugeordnet ist, kann eine Weiterverarbeitung der Anfrage vorgenommen werden. Nach dieser erfolgreichen Überprüfung wird das Behaviour SendRequestLinks

aktiviert. In der action-Methode dieses Behaviours wird eine ACL-Nachricht mit folgender Spezifikation konstruiert:

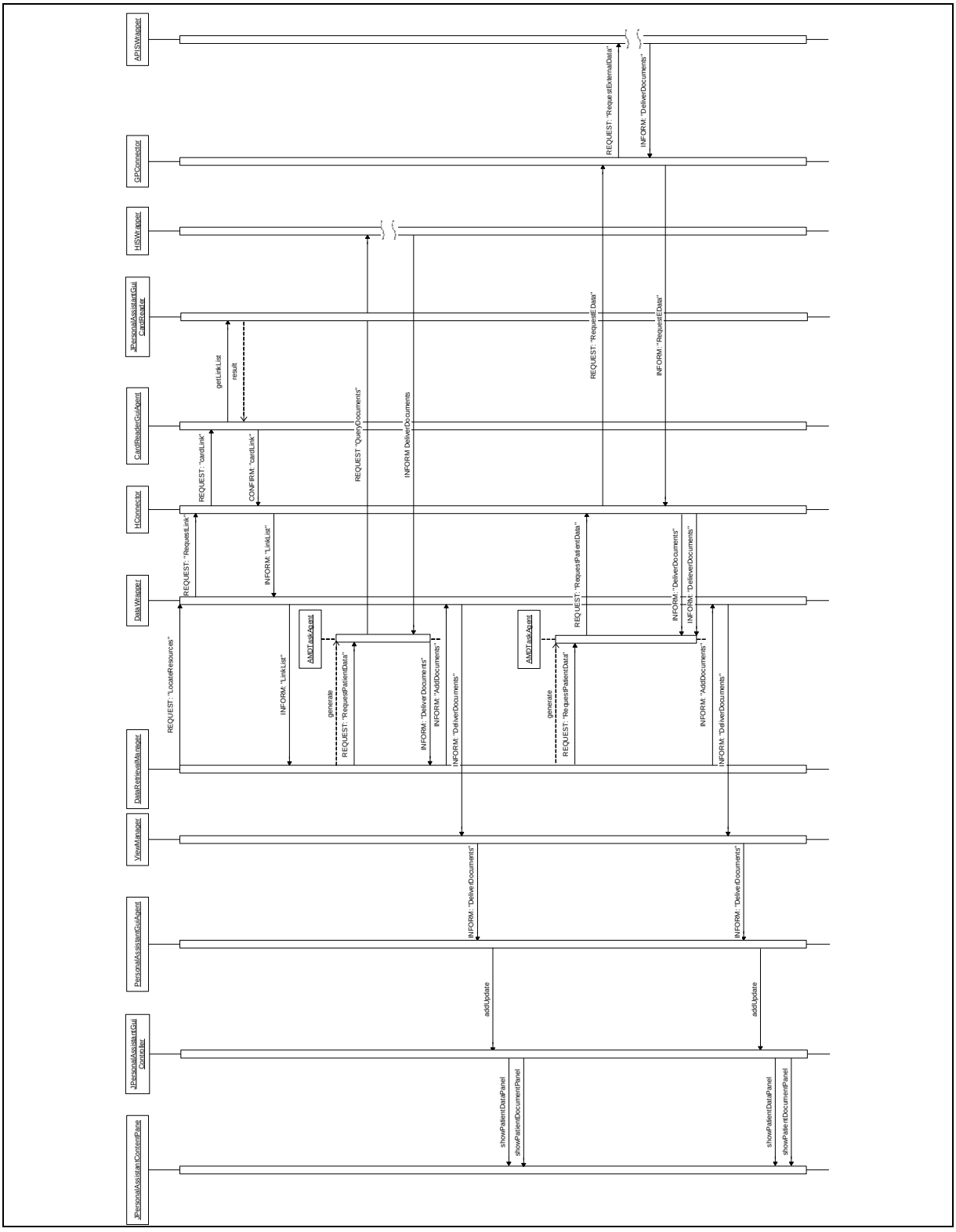

Abbildung 6.8-3: Interaktionen zum Abrufen der Patientendokumente
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

- Performativ REQUEST
- ConversationID RequestLink

Weiterhin wird das Ontologie-Objekt RequestLink zusammengesetzt und mit den Werten für Patientenidentifikation, Benutzerrolle sowie Benutzerlogin belegt und in die zu versendende Nachricht integriert. Zur Bestimmung des Empfängers dieser Nachricht wird über den Aufruf der Methode lookupConnector der Klasse DataWrapper der Konnektor mit dem Diensttyp „ConnectorData Service“ in der Umgebung Hospital gesucht. Der gefundene Konnektor, in diesem Fall der HConnectorAgent, wird als Ergebnis zurückgeliefert. Im weiteren Verlauf der action-Methode des Behaviours SendRequestLinks wird die konstruierte Nachricht an diesen Konnektor versandt. Dort wird die Behandlung der Nachricht vorgenommen (siehe dazu Abschnitt 6.8.3, Behaviour RequestLinkBehaviour).

In dem Behaviour ReceiveRequestLinks im DataWrapper wird die Antwort der soeben an den Konnektor versandten Anfrage verarbeitet. Dazu werden Nachrichten mit der folgenden Spezifikation erwartet:

- Performativ INFORM
- ConversationID LinkList

Die vom Konnektor empfangene Liste mit Verweisen zu den bisherigen Dokumenten des Patienten wird über das Behaviour SendLocateResources des Agenten DataWrapper an den DataRetrievalManager weitergeleitet. Dazu wird für den DataRetrievalManager die folgende Nachricht konstruiert und diese vom DataWrapper an ihn versandt:

- Performativ INFORM
- ConversationID LinkList

Im Behaviour ReceiveLocateRecources des Agenten DataRetrievalManager wird diese Nachricht verarbeitet. Darin wird in der zuvor gespeicherten Liste der Typen der angeforderten REQUESTs für die Dokumentenarten überprüft, ob der empfangene Dokumententyp von einem Agenten angefordert wurde. Ist dies der Fall, wird für jeden empfangenen Link über das Behaviour GenerateTaskAgent im DataRetrievalManager jeweils ein Task-Agent gestartet (Anforderung 18 aus Abschnitt 6.4.2). Mit dem Aufruf dieses Behaviours endet die Abarbeitung des Behaviours ReceiveLocateResources. Im Folgenden wird nun das Behaviour GenerateTaskAgent beschrieben:

In diesem Behaviour wird für jeden Link und den jeweiligen Typ des angeforderten Dokuments ein eindeutiger Task-Agent AMDTaskAgent gestartet. Für jeden dieser Agenten wird das Behaviour OnTaskAgentReady im DataRetrievalManager aktiviert. Darin wird im DataRetrievalManager auf Nachrichten der Agenten AMDTaskAgents mit der folgenden Spezifikation gewartet:

- Performativ INFORM
- ConversationID Spezifikation des jeweiligen AMDTaskAgents

Diese empfangenen Nachrichten zeigen an, dass die jeweiligen Task-Agenten vollständig gestartet sind. Sobald diese im DataRetrievalManager eintreffen, werden für jede eingehende Nachricht die beiden Behaviours ReceiveDeliverDocuments und SendRequestPatientData aktiviert. In dem Behaviour ReceiveDeliverDocuments wird für jeweils einen Task-Agenten seine Nachricht mit den eingehenden Dokumenten empfangen. Dazu wird auf Nachrichten mit der folgenden Spezifikation gewartet:

- Performativ INFORM
- ConversationID DeliverDocuments

Aus der empfangenen Nachricht werden die Dokumente aus der Ontologie DeliverDocuments entpackt und in das Ontologie-Objekt AddDocuments gepackt. Es folgt die Aktivierung des Behaviours SendAddNewDocuments, dem auch das Objekt AddDocuments übergeben wird. In diesem Behaviour wird vom DataRetrievalManager eine Nachricht an den DataWrapper mit der folgenden Spezifikation konstruiert, um die Dokumente an diesen weiterzuleiten:

- Performativ INFORM
- ConversationID AddDocuments

Diese Nachrichten werden vom DataWrapper im Behaviour ReceiveAddDocuments empfangen und weiterverarbeitet. Dazu wird zunächst auf die Nachrichten mit der oben angegeben Spezifikation INFORM und AddDocuments gewartet. Beim Eintreffen der Nachrichten wird überprüft, ob der sendende Agent, d.h. der DataRetrievalManager, mit dem DataWrapper interagieren darf. Dazu wird die Methode checkAccess aufgerufen. Bei erlaubtem Zugriff werden die Dokumente aus dem Ontologieobjekt AddDocuments entpackt und in das Ontologieobjekt DeliverDocuments gepackt[73]. Anschließend wird im Behaviour ReceiveAddDocuments die Aktivierung des Behaviours SendDeliverDocuments vorgenommen. Dieses leitet die empfangenen Dokumente an den ViewManager weiter. Im Behaviour SendDeliverDocuments wird eine Nachricht mit der folgenden Spezifikation generiert:

- Performativ INFORM
- ConversationID DeliverDocuments

Die empfangenen Dokumente werden schließlich über diese ACL-Nachricht an den ViewManager der Patientenakte weitergeleitet.

Im Behaviour ReceiveDeliverDocuments der Klasse ViewManager wird auf diese Nachrichten gewartet. In diesem Behaviour folgt die Aktivierung des Behaviours ProcessDataToGUI im ViewManager. In diesem Behaviour können die Daten für das jeweilige Anzeigegerät aufbereitet

[73] Für eine Performanzverbesserung ist eine persistente Speicherung der empfangenen Dokumente über die Aktivierung des Behaviours SaveTempRecord im Behaviour ReceiveAddDocuments vorgesehen. Die action-Methode dieses Behaviours ist jedoch in der vorliegenden Ausbaustufe noch nicht implementiert. Deshalb erfolgt aktuell keine lokale Speicherung der empfangenen Dokumente. Eine Implementierung der erforderlichen Funktionalitäten wird insgesamt aber vereinfacht und insbesondere die Integration des Konzepts zur Dokumentensynchronisation (siehe dazu Abschnitt 5.5.2) bei *Zachewitz/Schwolow* (2004) und *Zachewitz* (2004a; 2004b) ermöglicht.

werden[74]. Weiterhin werden die Dokumente in diesem Behaviour über die Aktivierung des Behaviours SendDeliverDocuments der Klasse ViewManager an den PersonalAssistantGuiAgent weitergeleitet. Dazu wird im Behaviour SendDeliverDocuments in der Klasse ViewManager eine Nachricht mit der folgenden Spezifikation gebildet:

- Performativ INFORM
- ConversationID DeliverDocuments

Diese Nachricht wird an den anfordernden Agenten gesendet, d.h. in diesem Fall an den PersonalAssistantGuiAgent. In dessen Behaviour ReceiveDeliverDocuments werden diese Nachrichten empfangen und weiterverarbeitet. In diesem Behaviour werden für die empfangenen Dokumente die Methode addUpdate der Klasse JPersonalAssistantGuiController aufgerufen. In dieser überladenen Methode wird je nach Art des Dokuments, Patientenstammdaten oder medizinische Dokumente, die Methode showPatientDataPanel bzw. showPatientDocumentPanel in der Klasse JPersonalAssistantContentPane aufgerufen. In diesen Methoden werden die erhaltenen Daten an der Benutzeroberfläche visualisiert.

In dem zuvor aktivierten Behaviour SendRequestPatientData im DataRetrievalManager (siehe dazu weiter oben) werden folgende Nachrichten zusammengestellt:

- Performative REQUEST
- ConversationID RequestPatientData

Diese Nachrichten werden vom DataRetrievalManager an den jeweiligen AMDTaskAgent gesendet. Sobald diese Task-Agenten vom Typ AMDTaskAgent gestartet werden (siehe dazu die Ausführungen weiter oben), werden in ihrer setup-Methode die folgenden Behaviours aktiviert:

- ReceiveRequestPatientData
- ReceiveDeliverDocuments

Sobald diese Behaviours aktiviert sind, wird eine INFORM-Nachricht an den initiierenden Agenten gesendet, d.h. den DataRetrievalManager. Sobald diese Nachricht beim DataRetrievalManager eingetroffen ist, können Nachrichten zum Abfragen der Patientendaten vom DataRetrievalManager an die Task-Agenten versandt werden, d.h. der wartende DataRetrievalManager wird über die Bereitschaft der Task-Agenten informiert. Das Behaviour ReceiveRequestPatientData eines AMDTaskAgents arbeitet Nachrichten mit der folgenden Spezifikation ab:

- Performativ REQUEST
- ConversationID RequestPatientData

In diesem Behaviour wird zunächst überprüft, ob die Daten aus einem lokalen oder einem entfernten Informationssystem extrahiert werden sollen. Erfolgt die Anfrage an ein lokales Informationssystem, wird das Behaviour SendQueryDocuments des AMDTaskAgents aktiviert. Sind

[74] Diese Funktionalität wird aktuell noch nicht implementiert, ist aber für eine Realisierung vorgesehen.

Daten aus einem entfernten Informationssystem abzufragen, wird das Behaviour SendRequest-PatientData des AMDTaskAgents aktiviert.

Im Behaviour SendQueryDocuments für aus Sicht des aktuellen Szenarios im KIS lokal gespeicherte Dokumente wird eine Nachricht mit der folgenden Spezifikation zusammengestellt:

- Performativ REQUEST
- ConversationID QueryDocuments

Diese Nachricht wird vom AMDTaskAgent an den entsprechenden Wrapper-Agenten gesendet, in diesem Fall ist dies der HISWrapper. Die Weiterverarbeitung im HISWrapper wird weiter unten beschrieben.

Im Behaviour SendRequestPatientData in der Klasse AMDTaskAgent für entfernt gespeicherte Dokumente wird zunächst eine Nachricht mit der folgenden Spezifikation konstruiert:

- Performativ REQUEST
- ConversationID RequestPatientData

Diese Nachricht wird an den entsprechenden Konnektor zur Weiterverarbeitung gesendet. In dem vorliegenden Fall ist dies der HConnector. Dort erfolgt die Verarbeitung dieser Anfrage. Im Behaviour SendRecieveExternalRequestBehaviour des HConnectors (Details zur Verarbeitung in diesem Behaviour sind in Abschnitt 6.8.3 beschrieben) wird die Anfrage an den externen Konnektor, d.h. für das vorliegende Szenario den GPConnector, weitergeleitet und dort im Behaviour ReceiveSendExternalRequestBehaviour empfangen. Diese Anfrage wird an den entsprechenden Wrapper-Agenten weitergeleitet, in diesem Fall ist dies der GPISWrapper (siehe dazu auch Anhang A.5.3). In dessen Behaviour ReceiveRequestExternalData werden Nachrichten mit der Spezifikation REQUEST und RequestExternalData verarbeitet. Dazu wird an dieser Stelle im Wrapper-Agenten GPISWrapper die Methode executeQuery aufgerufen. In dieser Methode werden die folgenden Schritte durchgeführt:

- Simulierter Zugriff auf die Daten in einem PVS: Auslesen eines lokal gespeicherten Arztbriefes im XML-Format und Transformation in ein CDA-Dokument mit LOINC-Codes (Anforderung 6 aus Abschnitt 6.4.1), Speicherung der Daten im Ontologieobjekt MedicalDocument (Anforderung 7 aus Abschnitt 6.4.1), Hinzufügen des Dokuments zu einer Liste von Dokumenten.
- Auslesen und Speichern eines Befundes analog zum Entlassbrief
- Verpacken der Liste der Dokumente im Ontologieobjekte DeliverDocuments
- Aktivierung des Behaviours SendDeliverDocuments

In dem Behaviour SendDeliverDocuments des GPISWrappers wird die folgende ACL-Nachricht konstruiert:

- Performativ INFORM
- ConversationID DeliverDocuments

Diese Nachricht wird mit den gefundenen medizinischen Dokumenten vom GPISWrapper an den GPConnector versandt. Dort wird diese Nachricht vom AnswerBehaviour des Behaviours ReceiveSendExternalRequestBehaviour empfangen und an den HConnector zurückgesendet. In dessen Behaviour AnsweringBehaviour des Behaviours SendReceiveExternalRequestBehaviour wird diese Nachricht empfangen und an den DataWrapper mit der ConversationID Deliver-Documents und dem Performativ INFORM weitergeleitet. Dort erfolgt die Weiterverarbeitung der empfangenen Dokumente, wie bereits weiter oben beschrieben.

Die Verarbeitung von Anfragen an den HISWrapper (für eine Beschreibung siehe auch Anhang A.5.4) umfassen die folgenden Schritte: Im Behaviour ReceiveQueryDocuments des HISWrapper wird die Nachricht REQUEST mit der ConversationID QueryDocuments empfangen. Sobald diese Nachricht eingetroffen ist, wird in der Klasse HISWrapper die Methode executeQuery ausgeführt. Darin werden die folgenden Schritte bearbeitet (Anforderung 8 aus Abschnitt 6.4.1):

- Öffnen einer JCo-Verbindung zum SAP IS-H- bzw. i.s.h.med-System über den Aufruf der Methode openDirectConnectionToSAP aus der Klasse SAPJCoConnection
- Wenn Befunde aus dem KIS extrahiert werden sollen, wird über den Aufruf search-PatientBAPI aus der Klasse SAPJCoConnection zunächst anhand einer externen Patientenidentifikation[75] der Patientenstammdatensatz aus dem SAP-System ausgelesen. Dazu wird die BAPI BAPI_PATIENT_SEARCH im SAP-System aufgerufen. Das zurück gelieferte Ergebnis wird über den Aufruf der Methode transformTableToXML aus der SAP-Tabellenstruktur in eine XML-Struktur überführt.
- Aufruf der Methode getInternalSAPiD in der Klasse HISWrapper: Dabei wird die interne Patientenidentifikation des Patienten innerhalb des KIS bzw. des Krankenhauses zurückgeliefert.
- Aufruf der Methode getPatientCaseListBAPI in der Klasse SAPJCoConnection: In dieser Methode wird die BAPI BAPI_PATIENT_GETCASELIST auf dem SAP-System aufgerufen. Der Rückgabewert ist die Fallliste für den aktuellen Patienten anhand der internen SAP-Patientenidentifikation.
- Aufruf der Methode getAllCaseids in der Klasse HISWrapper: Dabei wird eine Liste aller Fallidentifikationen zu dem aktuellen Patienten zurückgegeben.
- Für jeden dieser Identifikatoren wird die Methode getCaseFindingsListBAPI in der Klasse SAPJCoConnection aufgerufen. Für jeden Fall sollen über eine noch zu implementierende BAPI diejenigen Befunde zurückgeliefert werden, die zu einem bestimmten Patienten für einen Fall im SAP-System gespeichert sind. Weil diese BAPI in der der Implementierung zugrunde liegenden Version des SAP-Systems noch nicht realisiert ist, wird ihre Funktionalität simuliert, indem jeweils ein Link aus dem lokalen Dateisystem für alle Dateien mit der Spezifikation HIS/<Patientenidentifikation>/*.xml im Ergebnis der Anfrage hinterlegt wird.
- Aufruf der Methode getAllFindingsAsList in der Klasse HISWrapper: Für jeden Eintrag in der Befundliste wird über die Methode openPatientDataFromXMLFile der Inhalt aus der weiter oben spezifizierten Datei ausgelesen und über die Methode transformPatientData im AMDWrapperAgent in ein CDA-Dokument überführt. Es folgt eine entsprechende Be-

[75] Diese externe Identifikation entspricht derjenigen, die auch auf der eGK gespeichert ist, und damit idealerweise einer eindeutigen Patientennummer.

legung des Ontologieobjekts MedicalDocument, in dem die Patientenstammdaten und das CDA-Dokument gesetzt werden. Dieses Ontologieobjekt wird einer Liste hinzugefügt und als Rückgabewert der Methode getAllFindindsAsList zurückgeliefert.

- Dem Ontologie-Objekt DeliverDocuments wird das Ontologieobjekt MedicalDocument hinzugefügt. In letzterem befinden sich die eigentlichen Daten über den Patienten und seine medizinischen Dokumente.
- Abschließend wird das Behaviour SendDeliverDocuments für den HISWrapper aktiviert.

Im diesem Behaviour SendDeliverDocuments wird die folgende Nachricht konstruiert:

- Performativ INFORM
- ConversationID DeliverDocuments

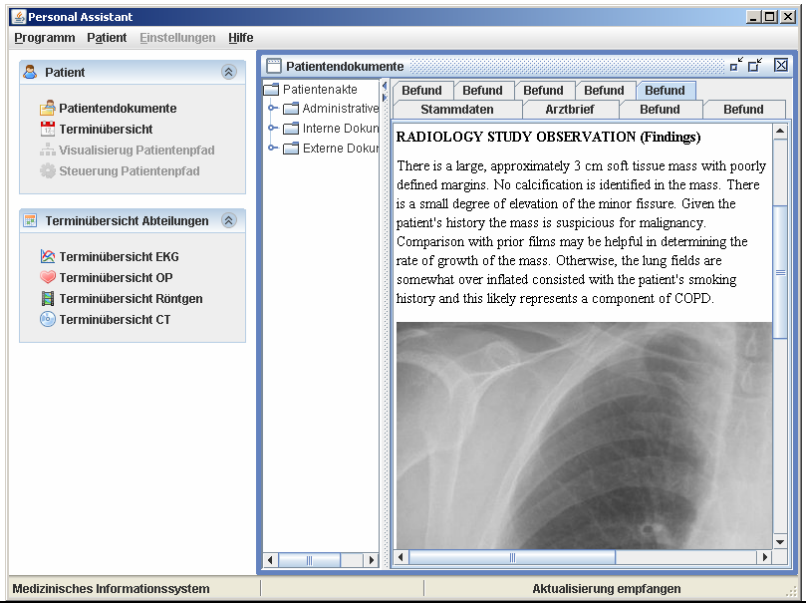

Abbildung 6.8-4: ***Desktop-Anwendung mit aggregierten Daten (Radiologiebefund[76])***
Quelle[77]: Eigene Darstellung

Diese Nachricht wird zusammen mit den extrahierten medizinischen Dokumenten an den anfragenden Agenten, d.h. den AMDTaskAgent, zurückgesendet. Im AMDTaskAgent wird im Behaviour SendDeliverDocuments diese Nachricht mit der Spezifikation INFORM und DeliverDocuments an den DataRetrievalManager weitergeleitet. Im DataRetrievalManager wird diese Nachricht im Behaviour ReceiveDeliverDocuments empfangen und anschließend das Behaviour SendAddNewDocuments aktiviert. Die weitere Verarbeitung erfolgt so, wie bereits weiter oben beschrieben.

---

[76] Die textuelle Beschreibung in diesem Screenshot ist dem HL7 CDA-Dokument „SINR CDA sample.xml“ entnommen, welches im Archiv 20070111_CDA_R2_examples.zip auf der folgenden Seite verfügbar ist: http://www.hl7.org, zugegriffen am 25.08.2007. Die Inhalte dieses Dokuments wurden im Rahmen der Arbeiten von *Bastian* (2005) und der vorliegenden Arbeit angepasst.

[77] Das in diesem Screenshot dargestellte Bildmaterial kann auf der folgenden Seite abgerufen werden: http://de.wikipedia.org/wiki/Bild:Thorax_pa_peripheres_Brronchialcarcinom_li_OF.jpg, zugegriffen am 18.08.2007. Dieses Dokument ist verfügbar unter der GNU Free Documentation License, Version 1.2, November 2002. Die zugehörigen Lizenzbestimmungen können auf der folgenden Seite abgerufen werden: http://www.gnu.org/licenses/fdl.html, zugegriffen am 25.08.2007.

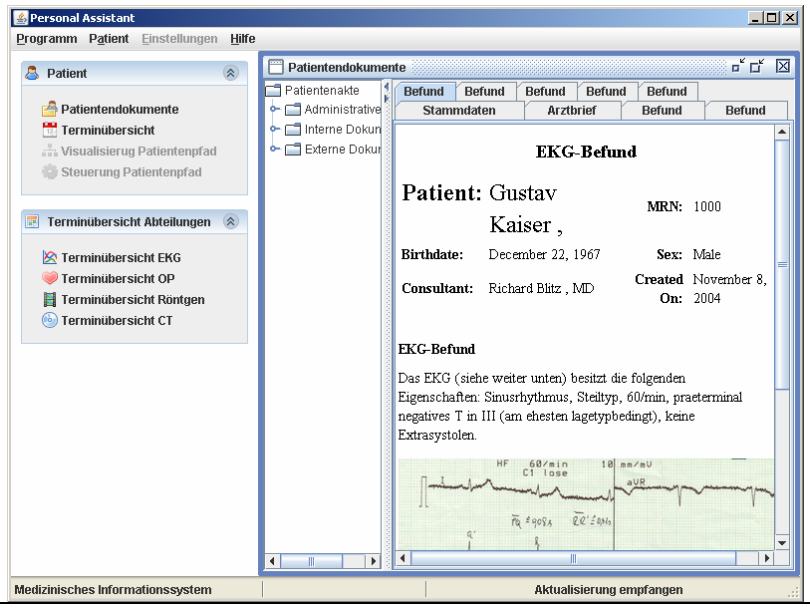

Abbildung 6.8-5: ***Desktop-Anwendung mit aggregierten Daten (EKG-Befund[78])***
Quelle[79]: Eigene Darstellung

Mit diesen Ausführungen sind die Verarbeitungsschritte von der Anforderung von Patientendokumenten aus verteilten Informationssystemen bis zu ihrer Visualisierung an der Benutzerschnittstelle der virtuellen Patientenakte beschrieben. Damit stehen dem Anwender alle Dokumente, die für einen Patienten in seiner bisherigen Krankheitsgeschichte hinterlegt wurden, für die aktuelle Entscheidungsfindung zur Verfügung, wie sie am Anfang dieses Abschnitts beschrieben wurde.

Die empfangenen Dokumente werden in einer Baumstruktur klassifiziert, um die Unterscheidung zwischen internen und externen Dokumenten zu ermöglichen. Diese Klassifikation ist aus Sicht des eingangs dieses Abschnitts beschriebenen Szenarios gestaltet. In Abbildung 6.8-4 und Abbildung 6.8-5 ist die virtuelle Patientenakte mit aggregierten Dokumenten dargestellt. Dabei ist insbesondere zu bemerken, dass über das Agentensystem auch multimediale Daten, in diesem Fall bspw. Bilddaten, transportiert werden können. Details dazu werden im folgenden Abschnitt 6.8.2 beschrieben.

Zusammenfassend kann mit der Beschreibung der Funktionalität zur Informationsaggregation festgestellt werden, dass über einen zentralen Eintrittspunkt in das AMD die medizinischen Dokumente des Patienten aggregiert werden können. Mit dieser Integration von Daten in dem AMD wird die Datenintegration entsprechend des EAI-Prinzips realisiert und damit Anforderung 15 aus Abschnitt 6.4.2 erfüllt. Außerdem wird mit der Trennung der Funktionalitäten in die Schichten der Benutzeroberfläche (z.B. ViewManager, JPersonalAssistantGui), des Ap-

[78] Die textuelle Beschreibung in diesem Screenshot ist dem HL7 CDA-Dokument „SINR CDA sample.xml“ entnommen, welches im Archiv 20070111_CDA_R2_examples.zip auf der folgenden Seite verfügbar ist: http://www.hl7.org, zugegriffen am 25.08.2007. Die Inhalte dieses Dokuments wurden im Rahmen der Arbeiten von *Bastian* (2005) und der vorliegenden Arbeit angepasst. Die Beschreibung des EKG-Befundes wurde von Herrn Dr. Max Dienel erstellt. Die persönlichen Daten des gezeigten Patienten sind fiktiv.

[79] Das in diesem Screenshot dargestellte Bildmaterial kann auf der folgenden Seite abgerufen werden: http://de.wikipedia.org/wiki/Bild:Standard-ekg.jpg, zugegriffen am 18.08.2007. Dieses Dokument ist verfügbar unter der GNU Free Documentation License, Version 1.2, November 2002. Diese Lizenz ist auf der folgenden Seite verfügbar: http://www.gnu.org/licenses/fdl.html, zugegriffen am 25.08.2007.

plikationskerns (z.B. AMDTaskAgent) und der Ressourcenintegration (z.B. HISWrapper, GPISWrapper) die agentenbasierte Anwendung in eine Schichtenarchitektur hineinentwickelt. Damit wird Anforderung 16 aus Abschnitt 6.4.2 erfüllt.

Der in diesem Abschnitt bereits erwähnte Versand von Bilddaten der virtuellen Patientenakte wird im Folgenden Abschnitt 6.8.2 beschrieben.

6.8.2 Einbettung von Bilddaten zum Versand über ACL-Nachrichten

Da sich Bild gebende Verfahren in der Medizin aufgrund neuer diagnostischer Verfahren verbreiten, ist auch der Versand von Bildmaterial in der prototypischen Implementierung geeignet zu berücksichtigen. Für den Versand von binären Bilddateien ist deshalb ein adäquates Verfahren zu entwickeln, das die Integration dieser Daten in das HL7 CDA-Format (siehe dazu Abschnitt 4.11.2.1.1) erlaubt, welches schließlich über eine ACL-Nachricht in einem Agentensystem versandt und auf der Empfängerseite korrekt interpretiert werden kann. Dazu sind die folgenden Schritte erforderlich, um binäre Bilddaten in das nonXMLBody-Element der CDA-Datei zu integrieren:

- Identifikation von Verweisen auf Bilddateien (gespeichert in den observationMedia-Elementen des CDA-Dokuments) in der HL7 CDA-Datei
- Einlesen der Bilddatei im Binärformat
- Codierung des Bildes im MIME-Code[80] (Multipurpose Internet Mail Extensions, Base64, zur Begründung der Auswahl dieses Codes siehe die Ausführungen weiter unten)
- Integration des Codes in das CDA-Dokument (siehe dazu auch Abbildung 6.8-6)
- Versenden des Dokuments in einer ACL-Nachricht
- Empfangen der ACL-Nachricht
- Decodieren der MIME-Codierung zum Binärformat
- Parsen des Bildnamens
- Lokale Speicherung des Bildes auf der Empfängerseite
- Visualisierung des Bildes über die Einbettung in ein HTML-Dokument

Auf der Wrapper-Seite der gekapselten Informationssysteme werden die Daten entsprechend der Beschreibung weiter oben codiert. Da diese Funktionalität mehrfach von spezialisierten Wrappern verwendet wird, wird diese von der Oberklasse AMDWrapperAgent an die jeweiligen Unterklassen, z.B. GPISWrapper oder HISWrapper, vererbt. In diesen Unterklassen werden in der Methode transformPatientData die Codierung und Einbettung der Bilddaten in dem Methodenaufruf encodeMedia der Klasse EncodeObservationMedia durchgeführt. Das so codierte Dokument kann in eine ACL-Nachricht integriert werden. Auf der Empfängerseite wird das Dokument von der Klasse PersonalAssistantGuiAgent entgegengenommen, genauer im Behaviour ReceiveDeliverDocuments, verarbeitet und über den Methodenaufruf decodeMedia in der Klasse DecodeObservationMedia für die Visualisierung an der Benutzeroberfläche aufbereitet. Vorteilhaft für diese Verarbeitung ist die Java Swing-Funktionalität (Anforderung 12 aus Abschnitt

[80] Weiterführende Informationen können auf den Seiten unter http://tools.ietf.org/html/rfc2045 bzw. http://tools.ietf.org/html/rfc2046#page-12, zugegriffen jeweils am 28.02.2007, abgerufen werden.

6.4.2), die die Einbettung von HTML-Code einschließlich Bilddaten in die Benutzeroberfläche erlaubt. Da die erhaltenen Bilder auf der Empfängerseite lokal gespeichert werden, werden diese vor dem Beenden der Anwendung bzw. beim Patientenwechsel wieder gelöscht.

```
<ClinicalDocument>
…
        <component>
                <structuredBody>
                …
                </structuredBody>
        </component>
        <nonXMLBody>
                <Radiologie[Patientenidentifikation]GPIS> …
                <EKG[Patientenidentifikation]HIS> …
        </nonXMLBody>
</ClinicalDocument>
```

Abbildung 6.8-6: ***HL7 CDA-Dokument mit codierten Bildern***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Die Ausführungen zeigen, dass über Agentennachrichten auch andere Daten als String- oder Ontologie-Objekte transferiert werden können. Dazu muss eine entsprechende Codierung erfolgen. Somit erhalten Agentensysteme einen wesentlichen Mehrwert hinsichtlich ihrer Praxistauglichkeit, weil auch Bildmaterial, eingebettet in medizinische Dokumente, übertragen und visualisiert werden kann. Exemplarisch wurde in der vorliegenden Implementierung die Integration von Bilddaten im Format JPEG demonstriert. Obwohl mit DICOM (siehe Abschnitt 4.11.2.1.2) ein etablierter Standard für die Übertragung und Speicherung von Bilddaten im medizinischen Bereich vorliegt, ist für die Ziele der prototypischen Implementierung in der vorliegenden Arbeit die Umsetzung anhand JPEG ausreichend. Diese Entscheidung wird mit dem folgenden Zusammenhang begründet:

Wilczek (2007, 232-235) stellt in seinem Ansatz eine Komponente zur Verfügung, die DICOM-Daten in das JPEG-Format transformiert. Diese Funktionalität kann in das vorliegende Agentensystem geeignet eingebettet werden: Entsprechend dem bei *Shaw/Garlan* (1996, 21-22) beschriebenen Architekturmuster „Pipes and Filters“ kann die Ausgabe der DICOM-JPEG-Transformation als Eingabe der JPEG-ACL-Transformation in der vorliegenden Arbeit dienen.

Abschließend kann für die Wahl des MIME-Formats festgestellt werden, dass über dieses Format auch weitere multimediale Daten wie Bewegtbilder oder Audiomaterial in Agentennachrichten eingebettet werden können. Somit stellt ein AMD auf der Basis von Software-Agenten ein geeignetes Medium dar, um unterschiedliche Datenformate zu transportieren.

6.8.3 Einbettung in die Telematikinfrastruktur

Ein Ziel der in dieser Arbeit beschriebenen Implementierung ist die Berücksichtigung von ausgewählten Sicherheitseigenschaften, welche durch die TI vorgegeben werden. Anforderungen zur Unterstützung dieser Charakteristika können wie folgt zusammengefasst werden:

- Auslesen von Patienten- und Anwenderdaten sowie von Verweislisten aus der eGK bzw. dem HBA über den Konnektor

- Überprüfung der Berechtigung zum Zugriff auf die Daten der eGK
- Konnektor als alleiniges Bindeglied zwischen einem Institut und der TI, Weiterleitung von Anfragen nach externen Patientendokumenten sowie Empfangen und Verarbeiten dieser Anfragen
- Protokollierung der 50 letzten Lesezugriffe auf die eGK
- Verschlüsseln und Signieren der ausgetauschten Daten

Im Folgenden wird beschrieben, wie diese Eigenschaften geeignet implementiert werden können. Als Grundlage für die Implementierung wird die JADE-S-Erweiterung (siehe dazu Abschnitt 6.3.4) vorausgesetzt (Anforderungen 19 und 20 aus Abschnitt 6.4.2). Die in diesem Abschnitt dargestellten Interaktionen zwischen den Agenten sind zusammengefasst aus der Darstellung bei *Haas* (2005b; 2005c). Letztere wird in das Szenario eingebettet, welches Teil des in der vorliegenden Arbeit als Anwendungsfall gewählten Prozesses der OP-Vorbereitung (Abschnitt 6.6.1) ist:

Anfragen des Anwenders an das AMD zur Datenaggregation werden zunächst an den Konnektor gerichtet. Dieser leitet die Anfrage zum Überprüfen der Zugriffsberechtigung und zum Auslesen der Daten der (simulierten) eGK an das Kartenlesegerät weiter. Wird der Zugriff erlaubt, können die Daten aus der eGK ausgelesen werden. Zugriffe auf die eGK zum Auslesen von Verweislisten oder zur Überprüfung der Zugriffsberechtigung werden in einer separaten Datei für jeden Patienten protokolliert. Dabei werden in einem zyklischen Verfahren die jeweils letzten 50 Zugriffe festgehalten, wie es auch in den Vorgaben für die TI festgelegt ist (siehe dazu die Ausführungen in Abschnitt 4.10.1.1). Festgeschrieben werden in der Protokolldatei exemplarisch das Login des Anwenders, die Datums- und Uhrzeitangabe sowie die Zugriffsart. Zur Reduktion der Implementierungskomplexität wird diese Protokollinformation in der vorliegenden Arbeit nicht entsprechend der Spezifikation für die TI auf einem zentralen (simulierten) Server der TI, sondern lediglich im lokalen Dateisystem abgelegt. Diese Implementierungsentscheidung reduziert die Qualität der Ergebnisse der vorliegenden Arbeit nicht, da eine Abbildung von Serverfunktionalitäten nicht im Fokus der vorliegenden Arbeit steht. Sobald die Verweisliste aus der eGK ausgelesen ist, wird ein Kommunikationskanal zwischen dem anfragenden Konnektor und dem entfernten Konnektor aufgebaut, über den die zu aggregierenden Daten ausgetauscht werden.

Weiterhin ist zu berücksichtigen, dass eine Institution jeweils durch einen eigenen Konnektor nach außen gekapselt wird. Nur diejenigen Dienste, die von extern in Anspruch genommen werden können, werden im DF der jeweiligen Plattform bzw. Umgebung registriert. So wird z.B. der Dienst, der die Anfragen anderer Teilnehmer der TI entgegennimmt und bearbeitet, als externer Dienst registriert. Die übrigen Dienste, die nicht von extern aufgerufen werden können, werden nicht im DF abgelegt.

Auf der Grundlage dieser Rahmenbedingungen werden im Folgenden die Interaktionen zwischen den an dem beschriebenen Szenario involvierten Agenten dargestellt. Diese Interaktionen werden jeweils aus der Sicht der Konnektoren HConnectorAgent (Plattform Hospital, Anforderung 1 aus Abschnitt 6.4.1) sowie GPConnectorAgent (Plattform General Practitioner, Anforderung 1 aus Abschnitt 6.4.1) und des Kartenlesers (CardReaderGuiAgent, Plattform Hospital,

Anforderung 2 aus Abschnitt 6.4.1) ausgeführt. Abgeschlossen wird diese Darstellung durch eine Zusammenfassung über die technisch möglichen Sicherheitsdienste.

Bei der Initialisierung der gesamten Applikation (siehe Abschnitt 6.8.1) werden in den Plattformen Hospital bzw. Telematics die Dienste „ConnectorLink Service“ und „ConnectorData Service“ bzw. „HConnectorExternal Service“ in den Directory Facilitators der jeweiligen Plattform registriert (für eine Beschreibung der Funktionalitäten dieser Dienste siehe Tabelle 6.8-3). Realisiert werden diese Schritte in der setup-Methode des Agenten HConnectorAgent. Um die durch die Erweiterung JADE-S ermöglichten Dienste nutzen zu können, wird in dieser Methode außerdem ein entsprechender SecurityHelper initialisiert. Die folgenden Behaviours zur Bearbeitung von Anfragen werden abschließend in der setup-Methode des Agenten HConnectorAgent aktiviert:

Dienst	Beschreibung	DF
HConnectorExternalService	Versand von extrahierten Informationen aus dem durch den Konnektor gekapselten Informationssystem an andere entsprechend anfragende Konnektoren	Telematics
ConnectorLink Service	Rückgabe der auf der eGK gespeicherten Verweisliste	Hospital
ConnectorData Service	Versand von Anfragen zur Extraktion von Daten an aus der Sicht des Konnektors externe Informationssysteme, die über die Einträge in der Verweisliste determiniert sind	Hospital

Tabelle 6.8-3: ***Überblick über registrierte Dienste im HConnectorAgent***
Quelle: Eigene Darstellung

- RequestLinkBehaviour: Versenden einer ACL-Nachricht an das Kartenlesegerät und Empfangen dieser Nachricht, Weiterleiten der erhaltenen Verweisliste an den anfragenden Agenten
- SendRecieveExternalRequestBehaviour: Empfangen einer Nachricht aus der eigenen Plattform zur Abfrage von Daten in einem aus Sicht des Konnektors externen Informationssystems, Weiterleitung dieser Nachricht an den Konnektor des externen Informationssystems
- ReceiveSendExternalRequestBehaviour: Empfangen einer Nachricht von einem externen Konnektor zur Abfrage von Daten in dem gekapselten Informationssystem, Weiterleitung der Anfrage an das aus Sicht des bearbeitenden Konnektors interne Informationssystem und Weiterleitung der von diesem Informationssystem erhaltenen Antwort an den ursprünglich anfragenden Konnektor
- CardReaderStatusRequestBehaviour: Empfangen einer Nachricht zur Weiterleitung an den Kartenleser, um in Erfahrung zu bringen, welche Karten aktuell eingelegt sind, Zusammenstellung einer Nachricht an den Sender der ursprünglichen Anfrage
- CheckAccessBehaviour: Empfangen einer Nachricht zur Weiterleitung an den Kartenleser, Überprüfung, ob der Zugriff berechtigt ist, Zurücksenden der Antwort an den ursprünglich anfragenden Agenten

Die Funktionalitäten der genannten Behaviours werden im Folgenden beschrieben: RequestLinkBehaviour erhält zur Abarbeitung der sicheren Kommunikation einen SecurityHelper und bearbeitet eingehende ACL-Nachrichten mit der folgenden Spezifikation:

- Performative REQUEST
- ConversationID RequestLink

Wird eine auf diese Art spezifizierte Nachricht empfangen, wird eine entsprechende an das (simulierte) Kartenlesegeräte zu versendende Nachricht konstruiert. Diese ACL-Nachricht erhält die folgende Spezifikation:

- Performative REQUEST
- ConversationID cardLink

Vor dem Versand dieser Anfrage an das Kartenlesegerät zum Auslesen der Verweisliste wird in der Klasse PProtocol (Anforderung 3 aus Abschnitt 6.4.1) die Methode protocolling aufgerufen. Darin wird der Lesezugriff auf die eGK des Patienten protokolliert. Dabei wird insbesondere gewährleistet, dass nur jeweils die letzten 50 Zugriffe auf die Karte gespeichert werden. Ist der Protokollvorgang erfolgreich, wird die Nachricht mit der oben genannten Spezifikation an das Kartenlesegerät versandt. Anschließend wird im Behaviour RequestLinkBehaviour auf eine entsprechende Antwort vom Kartenlesegerät mit der folgenden Spezifikation gewartet und diese empfangen:

- Performativ CONFIRM
- ConversationID cardLink

Treten beim Auslesen der Daten aus dem Kartenlesegerät Fehler auf, werden diese entsprechend ihrer Art verarbeitet. Weiterhin erhält der anfragende Agent vom Agenten HConnectorAgent eine entsprechende, in einer ACL-Nachricht verpackte Fehlermeldung. Können die Daten hingegen fehlerfrei aus dem Kartenlesegerät gelesen werden, sendet der Agent HConnectorAgent eine Nachricht an den anfragenden Agenten mit dem Performativ INFORM. Die Abarbeitung des Behaviours RequestLinkBehaviour für die Anforderung einer Verweisliste ist damit abgeschlossen.

Das Behaviour SendReceiveExternalRequestBehaviour versendet Anfragen an einen externen Konnektor, um die Daten aus externen Informationssystemen zu aggregieren. Dieses Behaviour erhält zur Abarbeitung der sicheren Kommunikation einen SecurityHelper. Damit wird ermöglicht, die von JADE-S angebotenen Dienste einzusetzen. Das Behaviour arbeitet eingehende Nachrichten mit der folgenden Spezifikation ab:

- Performativ REQUEST
- ConversationID RequestPatientData

Wird eine solche ACL-Nachricht empfangen, wird zunächst eine Antwortnachricht mit der folgenden Spezifikation vorbereitet:

- ConversationID DeliverDocuments

- Setzen des InReplyTo-Felds aus dem InReplyTo-Feld der empfangenen Nachricht

Sobald überprüft ist, ob in der empfangenen Nachricht der Inhalt RequestPatientData enthalten ist, wird die Adresse des externen Konnektors in der Umgebung Telematics gesucht. In dem vorliegenden Szenario ist dies der GPConnectorAgent. Das Objekt RequestPatientData spezifiziert dazu, wo die gewünschten Informationen zu lokalisieren sind. Die Nachricht an diesen Konnektor besitzt die folgende Spezifikation:

- Performativ REQUEST
- ConversationID RequestEData

Diese Nachricht wird nun nach dem Signieren durch den HConnectorAgent an den externen Konnektor, d.h. im beschriebenen Szenario an den GPConnectorAgent gesendet. Weiterhin wird der Filter für die daraufhin von diesem Konnektor zu empfangende Nachricht mit der folgenden Spezifikation gesetzt:

- ConversationID RequestEData
- Das InReplyTo-Feld wird mit dem Feld aus ReplyWith der an den Konnektor versandten Nachricht gesetzt.

Zum Empfangen der Antwort auf die an den externen Konnektor versandte Nachricht wird das Behaviour AnsweringBehaviour aktiviert. Wenn die oben spezifizierte Nachricht des externen Konnektors mit dem Performativ INFORM empfangen wird, wird der Nachrichteninhalt entpackt und vom Agenten HConnectorAgent an den originär anfragenden Agenten zurückgesendet. Auch im Behaviour SendReceiveExternalRequestBehaviour werden möglicherweise auftretende Fehler geeignet behandelt.

Das Behaviour ReceiveSendExternalRequestBehaviour arbeitet Anfragen eines externen Konnektors ab, um Daten aus dem aus der Sicht des Konnektors, der die Nachricht empfängt, internen Informationssystem zu aggregieren. Diesem wird zur sicheren Kommunikation ein SecurityHelper übergeben und er verarbeitet eingehende Nachrichten mit der folgenden Spezifikation:

- Performativ REQUEST
- ConversationID RequestEData

In dem empfangenen Objekt RequestExternalData solcher ACL-Nachrichten ist der Wrapper-Agent spezifiziert, an den die Anfrage zur Extraktion der dort gespeicherten Daten versandt werden soll. Der Dienst dieses Agenten wird im internen DF des empfangenden Konnektors gesucht. In dem beschriebenen Szenario ist dies der DF der Umgebung General Practitioner. An diesen Wrapper-Agenten wird eine Nachricht gesendet, die wie folgt spezifiziert ist:

- Performativ REQUEST
- ConversationID RequestExternalData

Vor der Verarbeitung zum Empfang der Nachricht vom Wrapper-Agenten wird das entsprechende Template spezifiziert:

- ConversationID DeliverDocuments
- InReplyTo wird mit dem InReplyTo-Feld aus der Nachricht an den Wrapper-Agenten besetzt

Zum Empfangen dieser Nachricht wird das Behaviour AnswerBehaviour aktiviert. Wird die spezifizierte Nachricht empfangen, wird die weiterzuleitende Nachricht vorbereitet. Dazu wird in dieser das Performativ INFORM gesetzt und der vom Wrapper-Agenten in der Nachricht mitgeschickte Inhalt (DeliverDocuments) übergeben. Diese Nachricht wird signiert und schließlich an den ursprünglich anfragenden Agenten versandt. Mit diesem Schritt ist das Behaviour AnswerBehaviour bzw. auch ReceiveSendExternalRequestBehaviour in seiner Bearbeitung abgeschlossen.

Mit dem Behaviour CardReaderStatusRequestBehaviour wird vom HConnectorAgent eine Anfrage an den Kartenleser gesendet, um in Erfahrung zu bringen, welche Karten mit welchen Daten aktuell eingelegt sind. Nach der Initialisierung dieses Behaviours wird der Filter für die zu empfangende Nachricht wie folgt spezifiziert:

- Performativ REQUEST
- ConversationID get CardReaderStatus

Nach dem Empfang einer solchen Nachricht wird eine Antwortnachricht an den anfragenden Agenten mit der folgenden Spezifikation zusammengestellt:

- Performativ INFORM
- ConversationID send CardReaderStatus

Nach der Suche nach dem Kartenleser (CardReaderGuiAgent) in der Umgebung Hospital wird eine entsprechende Nachricht an diesen vorbereitet. Dazu wird eine ACL-Nachricht mit dem Performativ REQUEST und der ConversationID get CardReaderStatus generiert und an den Kartenleser versandt. Als Antwort vom Kartenleser wird nun eine Nachricht mit der folgenden Spezifikation erwartet:

- Performativ INFORM
- ConversationID send CardReaderStatus

Der Inhalt dieser vom Kartenleser empfangenen Nachricht wird entpackt und in die Antwortnachricht an den ursprünglich anfragenden Agenten gepackt. Diese Nachricht erhält das Performativ INFORM und die ConversationID send CardReaderStatus. Sobald die Nachricht an den ursprünglich anfragenden Agenten versandt ist, ist das Behaviour CardReaderStatusRequestBehaviour abgearbeitet.

Im CheckAccessBehaviour wird überprüft, ob ein Zugriff auf die aktuell eingelegten Karten zulässig ist. In der action-Methode dieses Behaviours wird der folgende Filter für eingehende Nachrichten gesetzt:

- Performativ REQUEST
- ConversationID CheckAccess

Sobald eine solche Nachricht eingeht, wird in dem gewählten Szenario zunächst der Kartenleser in der Umgebung Hospital gesucht. Es folgt eine Zusammenstellung einer Nachricht an diesen Kartenleser mit der folgenden Spezifikation:

- Performativ REQUEST
- ConversationID checkAccess

Diese Nachricht wird nun an den Kartenleser versandt und auf die entsprechende Antwort des Kartenlesers mit dem Performativ CONFIRM und der ConversationID checkAccess gewartet. Sobald die erwartete Nachricht vom Kartenleser eintrifft, wird die folgende Nachricht als Antwort an den ursprünglich anfragenden Agenten generiert:

- Performativ CONFIRM
- Inhalt „Zugangsdaten korrekt“

Nach dem Versand dieser Nachricht ist das Behaviour CheckAccessBehaviour abgearbeitet. Auch in diesem Behaviour wird die Bearbeitung von Fehlerfällen unterstützt.

Bei der Initialisierung des Agenten GPConnectorAgent werden in seiner setup-Methode die Dienste ConnectorLink Service und ConnectorData Service in der Umgebung General Practitioner bzw. PConnectorExternal Service in der Umgebung Telematics registriert. Außerdem erfolgt hier eine Initialisierung des SecurityHelpers, um den Einsatz von Sicherheitsmechanismen der JADE-S-Erweiterung zu ermöglichen. Im Rahmen der Initialisierung folgt abschließend die Aktivierung der folgenden Behaviours:

- RequestLinkBehaviour
- SendReceiveExternalRequestBehaviour
- ReceiveSendExternalRequestBehaviour

Die Verarbeitung dieser Behaviours entspricht derjenigen, die in diesem Abschnitt für den HConnectorAgent weiter oben beschrieben wurde. Mit der Auslagerung der Behaviours und ihrem Einsatz in mehreren Agenten wird die Wiederverwendung der genannten Module demonstriert.

Der Agent CardReaderGuiAgent (für eine Übersicht über die Verarbeitungsschritte siehe auch Anhang A.5.8) repräsentiert diejenige Funktionalität, die vom Kartenleser zur Verfügung gestellt wird. Zur Initialisierung wird in seiner setup-Methode der Dienst „CardReader GUI Service“ in der Umgebung Hospital registriert. In seinen Behaviours wird auf die in Tabelle 6.8-4 beschriebenen Nachrichten reagiert und diese werden auf die dort dargestellte Weise verarbeitet.

Nachrichtenspezifikation	Verarbeitung der Nachricht	Antwort auf die eingehende Nachricht
• Performativ REQUEST • ConversationID cardLink	Auslesen der Verweisliste aus der eGK	• Performativ CONFIRM • Versand der Antwort mit den Verweisen an den anfragenden Agenten
• Performativ REQUEST • ConversationID „get CardReaderStatus"	Auslesen der Daten aus der eGK und dem HBA	• Performativ INFORM • ConversationID „send CardReaderStatus" • Versand der Antwort mit dem Patientennamen, der -identifikation, der Benutzerrolle und dem -namen an den anfragenden Agenten
• Performativ REQUEST • ConversationID checkAccess	Überprüfung der in der Anfrage überlieferten Benutzer- und Patientendaten mit denjenigen, die auf den eingelegten Karten gespeichert sind	• Performativ CONFIRM • Versand der Antwort an den anfragenden Agenten

Tabelle 6.8-4: ***Überblick über im CardReaderGuiAgent verarbeitete Nachrichten***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Entgegen dem in Abschnitt 6.7.2 dargestellten Design konnten in der vorliegenden Implementierung nicht alle geforderten Funktionalitäten zur Gewährleistung der Datenintegrität und -vertraulichkeit umgesetzt werden. Trotz der Beschreibung durch das *JADE Board* (2005, 6), nach der Nachrichten auch zwischen verschiedenen Plattformen verschlüsselt und signiert ausgetauscht werden können, zeigte sich in der prototypischen Implementierung, dass in der aktuellen Version von JADE-S nur die Signatur zwischen zwei Plattformen unterstützt wird. Im Gegensatz dazu konnte zwischen zwei verschiedenen Plattformen die Verschlüsselung von Nachrichten nicht erfolgreich umgesetzt werden. In Abbildung 6.8-7 ist die implementierte Verschlüsselung bzw. Signatur von Nachrichten entsprechend gekennzeichnet.

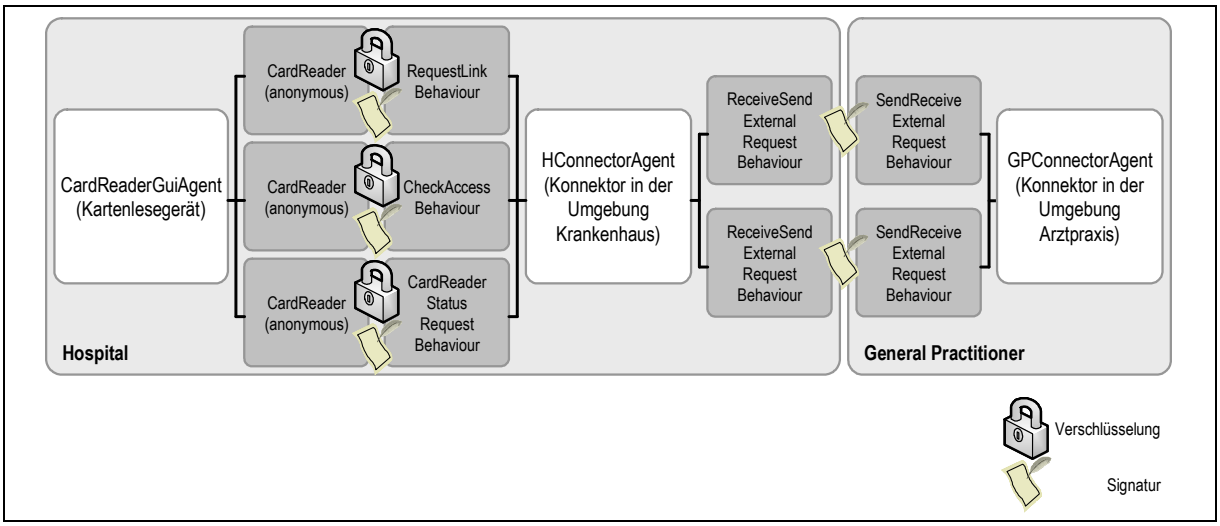

Abbildung 6.8-7: ***Implementierung Verschlüsselung und Signatur***
Quelle: Eigene Darstellung

Die Ausführungen zur Umsetzung von ausgewählten Sicherheitseigenschaften durch JADE-S zeigen zwar, dass diese nur eingeschränkt implementiert werden können. Aus technischer Perspektive lässt sich aber keine Be-

gründung ableiten, die eine vollständige Realisierung der Mechanismen zur Verschlüsselung und Signatur ausschließen würde. Bei entsprechender Weiterentwicklung von JADE-S kann diese Komponente deshalb zur Erfüllung der Sicherheitseigenschaften zur Verschlüsselung und Signatur in einem agentenbasierten System eingesetzt werden.

6.8.4 Mobile Anwendung

Das in Abschnitt 6.7.3 beschriebene Design für die mobile Anwendung wird in diesem Abschnitt in seinen Implementierungsdetails auf der Basis von JADE-LEAP (Anforderung 19 aus Abschnitt 6.4.2) beschrieben. Diese Ausführungen fassen die Darstellung bei *Hillebrand* (2006, 75-83) zusammen und werden entsprechend der Implementierung in der vorliegenden Arbeit erweitert. Fokussiert wird dabei die Beschreibung der Interaktionsmodellierung.

Bei der folgenden Darstellung wird vorausgesetzt, dass in der Desktop-Anwendung die erforderlichen Karten (eGK und HBA) bereits eingelegt sind. Die TCP/IP-Verbindung zwischen der mobilen Anwendung und dem Kartenleser ist durch eine drahtlose Netzwerkverbindung bereits gegeben. Im Folgenden werden darauf aufbauend die Interaktionen für die Push- und Pull-Funktionalität zur Zusammenstellung der virtuellen Patientenakte beschrieben:

Bei der Initialisierung der mobilen Anwendung wird der Agent mit dem Namen PersonalAssistant als Instanz der Klasse PersonalAssistantGuiAgent (für eine Darstellung der Verarbeitungsvorschrift siehe Anhang A.5.10) gestartet. In dessen Konstruktor wird zunächst der Konstruktor der Oberklasse AMDPersonalAssistantAgent aufgerufen. Darin werden die zu verwendende Ontologie und der Codec registriert (Hillebrand 2006, 76). Nach der Abarbeitung dieses Konstruktors wird die Initialisierung in der setup-Methode des Agenten PersonalAssistantGuiAgent (Anforderung 13 aus Abschnitt 6.4.2) fortgeführt. Darin erfolgt zunächst eine Registrierung dieses Agenten im DF der Hospital-Umgebung. Weiterhin wird eine Instanz der Klasse PersonalAssistantGui (Anforderung 13 aus Abschnitt 6.4.2) gebildet. In deren Konstruktor wird die Benutzeroberfläche zusammengestellt. Dafür werden auf der Basis der Bibliothek Java AWT (Anforderung 14 aus Abschnitt 6.4.2) die folgenden grafischen Elemente generiert:

- Hauptfenster
- Menüeinträge
- Bestätigungsdialoge

Mit dem Aufruf der Methode showLoginDialog wird die Abarbeitung des Konstruktors beendet. In dieser Methode wird das Login-Fenster (siehe Abbildung 6.8-8) konstruiert und zur Anzeige gebracht. Nach der Eingabe des Benutzernamens und des Logins wird über einen Klick auf den Button OK die Methode checkLogin über den entsprechenden Listener aufgerufen. Nach erfolgreicher Überprüfung der Eingabewerte wird einerseits das Behaviour ExternalCallListener dem Agenten PersonalAssistantGuiAgent hinzugefügt. Mit diesem Verhalten wird auf externe Push-Befehle von anderen Anwendungen (z.B. der Desktop-Applikation) reagiert, die die Übertragung der aggregierten Informationen über den gewählten Patienten auf das mobile Endgerät anstoßen. Andererseits wird die Methode showControlCenter aufgerufen. In dieser Methode werden – in Abhängigkeit des Status der Applikation – die folgenden Elemente der Benutzeroberfläche (siehe Abbildung 6.8-9) generiert und schließlich visualisiert:

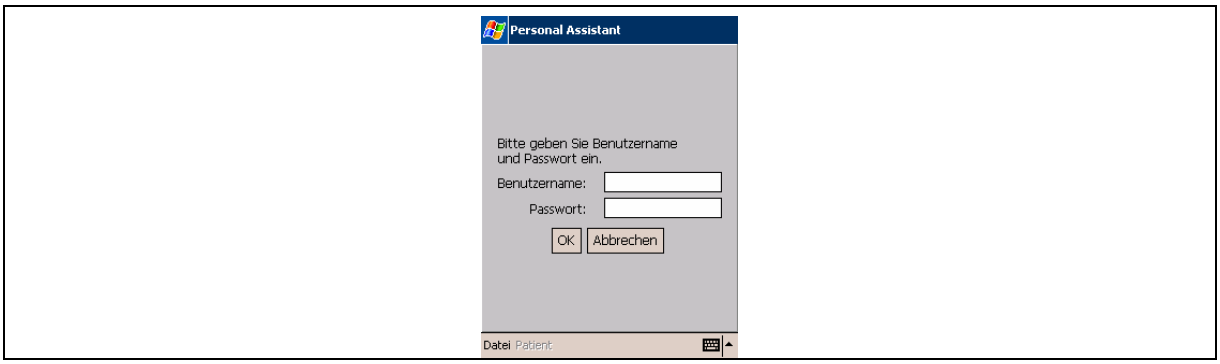

Abbildung 6.8-8: ***Login-Fenster der mobilen Anwendung***
Quelle: Eigene Darstellung

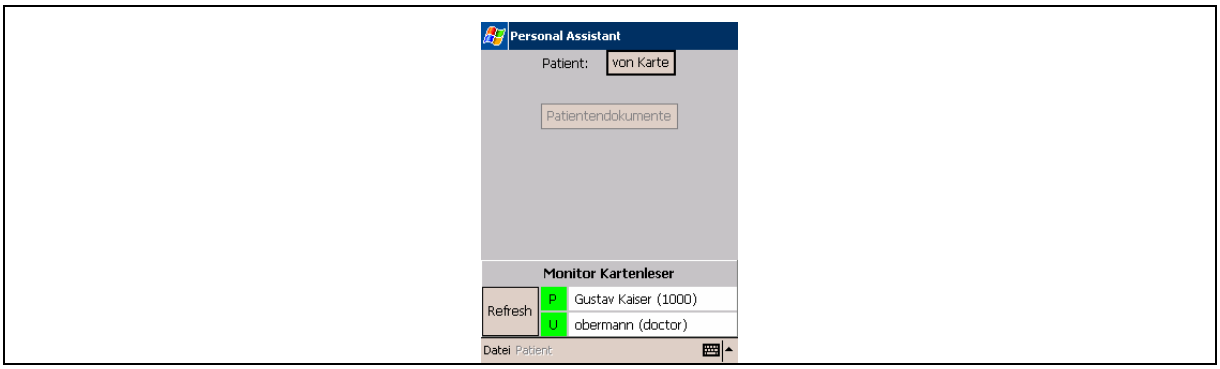

Abbildung 6.8-9: ***Hauptfenster der mobilen Applikation***
Quelle: Eigene Darstellung

- Button zum Auslesen der Patientendaten aus der eingelegten Patientenkarte
- Anzeige der aktuellen Patientenidentifikation
- Button zum Abrufen der Dokumente für den gewählten Patienten
- Monitor zur Anzeige der Verbindung zum in der Desktop-Anwendung simulierten Kartenlesegerät

Abschließend wird in der Methode showControlCenter die Methode getCardReaderStatus des Agenten PersonalAssistantGuiAgent aufgerufen, um die Daten aus den im Kartenleser eingelegten Karten abzufragen. Die nun folgenden Interaktionen laufen zwischen dem PersonalAssistantGuiAgent und der Desktop-Plattform ab und sind in Abbildung 6.8-10 in einem Sequenzdiagramm zusammengefasst. Das diese Interaktion ausführende Behaviour des Agenten PersonalAssistantGuiAgent ist in der Klasse GetCardReaderStatus definiert und wird im Aufruf der Methode getCardReaderStatus aktiviert.

Im Konstruktor dieses Behaviours wird zunächst im DF der Plattform Telematics nach dem HConnectorAgent gesucht, um dessen AID in Erfahrung zu bringen. Dazu ist es erforderlich, die Adresse des DF aus der Klasse Environment auszulesen (Hillebrand 2006, 78). Sobald der AID des HConnectorAgents bekannt ist, wird in der action-Methode des Behaviours GetCardReaderStatus an den HConnctorAgent eine Nachricht mit den folgenden Eigenschaften versandt:

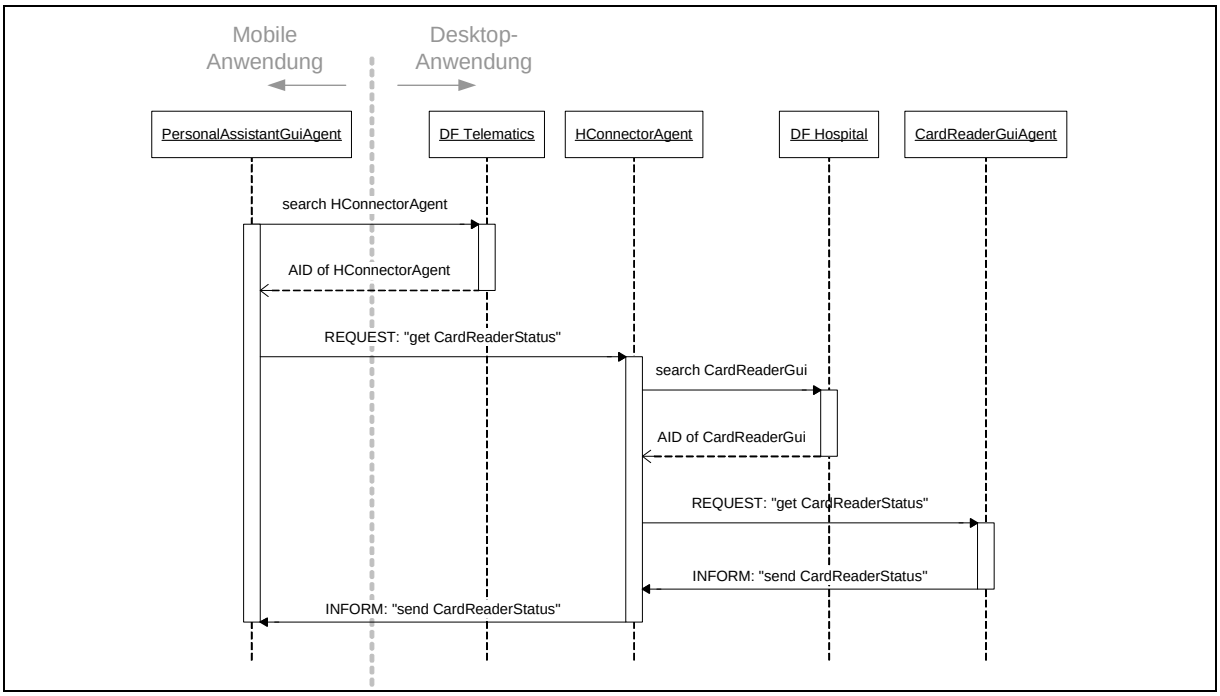

Abbildung 6.8-10: Interaktionen der mobilen Anwendung mit dem Kartenlesegerät
Quelle: In Anlehnung an *Hillebrand* (2006, 78)

- Performativ REQUEST
- ConversationID „get CardReaderStatus“

Mit der Weiterverarbeitung dieser Anfrage wird im HConnectorAgent das Behaviour CardReaderStatusRequestBehaviour beauftragt. Darin sucht der HConnectorAgent zunächst nach dem AID für die Benutzeroberfläche des Kartenlesers der Desktop-Anwendung im DF der Umgebung Hospital. Sobald der AID zur Verfügung steht, wird die Anfrage REQUEST mit der ConversationID „get CardReaderRequest“ an den CardReaderGuiAgent gestellt. Dieser nimmt die Anfrage in seiner action-Methode entgegen und sendet in der Nachricht mit der ConversationID „send CardReaderStatus“ die folgenden Informationen, die aus den eingelegten Karten über die Klasse JPersonalAssistantGuiCardReader ausgelesen werden:

- Name und Identifikation des Patienten (eGK)
- Name und Rolle des Benutzers (HBA)

Vor dem Versand dieser Informationen werden diese in dem Ontologie-Objekt CardReaderStatus abgelegt. Die Nachricht wird schließlich über den HConnectorAgent an den PersonalAssistantGuiAgent der mobilen Anwendung zurückgesandt. In dessen Behaviour GetCardReaderStatus wird der Inhalt der Nachricht entpackt, insbesondere werden die gewünschten Daten in einem temporären Objekt gespeichert. Über die Methode createCardReaderMonitor in der Klasse PersonalAssistantGui werden die Daten zur Weiterverarbeitung vom CardReaderGuiAgent an die Benutzeroberfläche PersonalAssistantGui übergeben. Trifft innerhalb einer zeitlichen Schranke keine Antwort vom HConnectorAgent an den PersonalAssistantGuiAgent ein, wird die Fehlermeldung „connection timeout“ an der Benutzeroberfläche PersonalAssistantGui angezeigt.

Im Behaviour GetCardReaderStatus des Agenten PersonalAssistantGuiAgent wird abschließend die Methode createCardReaderMonitor der Klasse PersonalAssistantGui aufgerufen. Darin werden die Daten zur Anzeige an der Benutzeroberfläche aufbereitet. Können Patient und Anwender nicht identifiziert werden, wird an der Benutzeroberfläche eine entsprechende Meldung angezeigt. Konnten die Daten hingegen erfolgreich aus den im Kartenleser eingelegten Karten gelesen werden, werden diese an der Benutzeroberfläche visualisiert (siehe Abbildung 6.8-9).

Die Identifikation des Patienten wird bereits beim Auslesen der Daten aus dem Kartenleser nach dem beschriebenen Vorgang ausgelesen und gespeichert sowie nach dem Klick auf den Button von Karte an der Benutzeroberfläche angezeigt (siehe Abbildung 6.8-11). Für die Zusammenstellung der virtuellen Patientenakte nach der Pull-Funktionalität werden die folgenden Schritte (siehe Abbildung 6.8-12) durchlaufen (Hillebrand 2006, 79-80):

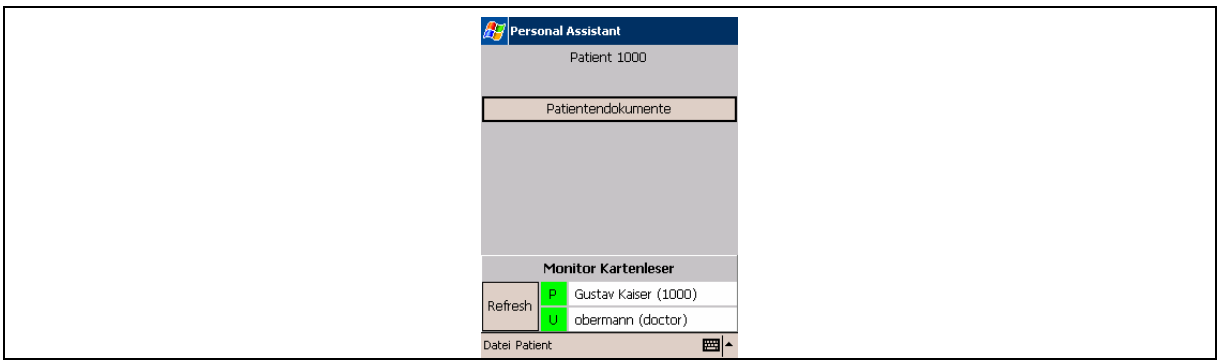

Abbildung 6.8-11: ***Mobile Applikation nach der Übernahme der Patientenidentifikation***
Quelle: Eigene Darstellung

1. Wird über den Button Patientendokumente auf der Benutzeroberfläche der Klasse PersonalAssistantGui die Zusammenstellung der virtuellen Patientenakte angestoßen, wird die Methode setNewPatientId über den Listener des Buttons aufgerufen. In dieser Methode wird die aus der Klasse AMDPersonalAssistantAgent vererbte Methode lookUpEHR der Klasse PersonalAssistantGuiAgent aufgerufen. In letzterer wird der AID des DataRetrievalManagers des für den Patienten relevanten AMD vom DF der Umgebung Hospital abgefragt. Das Ergebnis der Suche wird an die Methode setNewPatientId zurückgegeben, der AID wird in der Klasse AMDPersonalAssistantAgent gespeichert. Das Ergebnis der Suche kann entweder den Erfolg oder eine Fehlermeldung anzeigen. Die Erfolgsmeldung gibt an, dass der DataRetrievalManager und damit auch das zugehörige AMD des Patienten im DF bekannt sind. Die Fehlermeldung beschreibt, dass das AMD des Patienten nicht gefunden werden konnte.
2. Im Erfolgsfall des Aufrufs der Methode setNewPatientId wird in der Klasse PersonalAssistantGui im Agenten PersonalAssistantGuiAgent die Methode sendQueryPatientData aufgerufen. Andernfalls wird keine weitere Verarbeitung des ursprünglichen Aufrufs vorgenommen. In der Methode sendQueryPatientData wird das Behaviour SendQueryPatientData im Agenten PersonalAssistantGuiAgent aktiviert. Konnte durch den Aufruf der Methode lookUpEHR der DataRetrievalManager des zugehörigen AMD des Patienten gefunden werden (siehe Schritt 1), wird in der action-Methode des Behaviours SendQueryPatientData die Methode sendQuery2AMD aufgerufen. In dieser Methode wird eine ACL-Nachricht mit dem Performativ REQUEST und der ConversationID QueryPatientData

konstruiert und an den DataRetrievalManager des AMD des Patienten versandt. Die weiteren Verarbeitungsschritte sind Details der Desktop-Anwendung, werden im Abschnitt 6.8.1 ausführlich beschrieben und sind deshalb in Abbildung 6.8-12 nicht dargestellt.

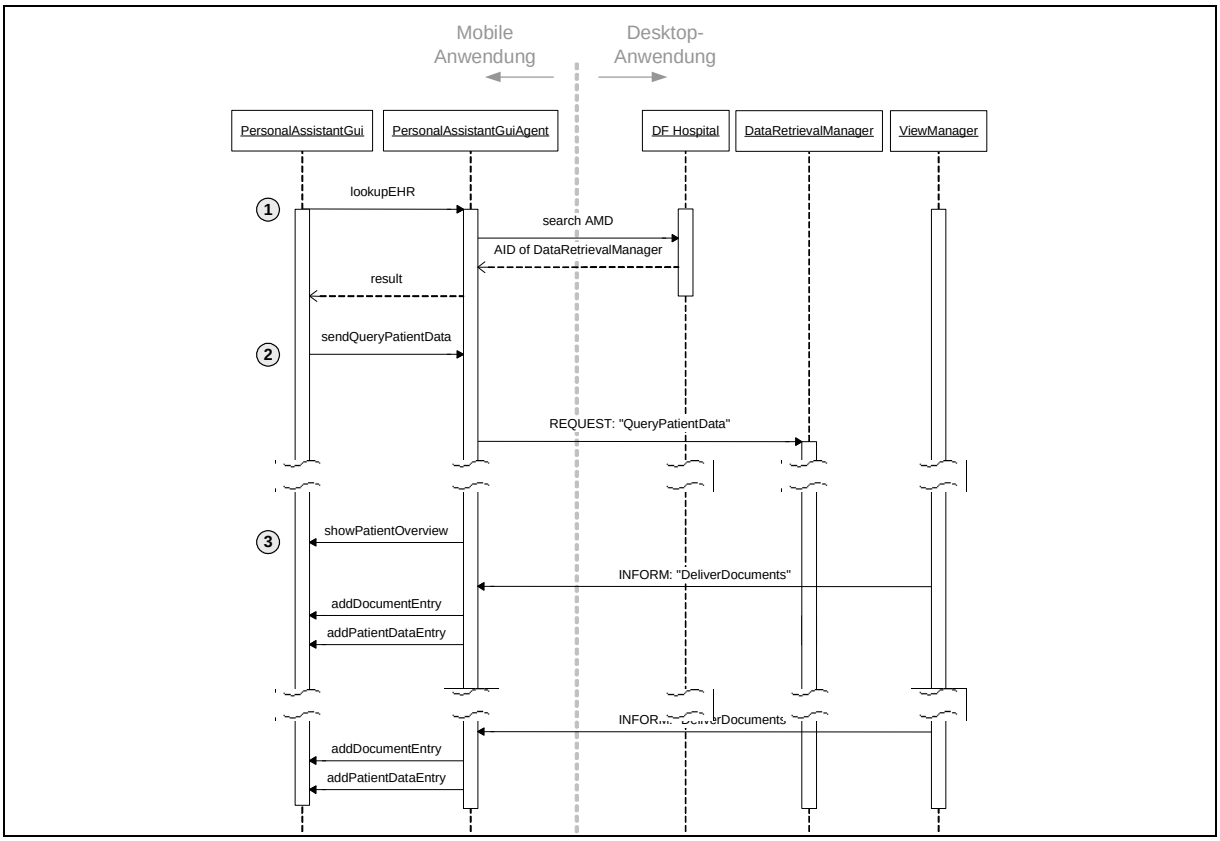

Abbildung 6.8-12: Interaktionen der mobilen Anwendung mit dem AMD
Quelle: In Anlehnung an *Hillebrand* (2006, 79)

3. Sobald diese Suchanfrage abgeschickt ist, wird in der Methode sendQuery2AMD die Methode showPatientOverview der Klasse PersonalAssistantGui aufgerufen. In dieser Methode wird ein Fenster zusammengestellt, in dem die erhaltenen Dokumente zu dem gewählten Patienten gelistet werden sollen. Darin werden die Dokumente entsprechend ihrem Herkunftsort nach intern (Daten aus denjenigen Informationssystemen, die der Institution zugeordnet sind, welche die Zusammenstellung der virtuellen Patientenakte anforderte) und extern (Daten aus entfernten Informationssystemen) kategorisiert. Administrative Daten werden gesondert dargestellt. Zum Empfang der angeforderten Dokumente wird in der Methode sendQuery2AMD abschließend das Behaviour ReceiveDeliverDocuments im PersonalAssistantGuiAgent aktiviert. Wenn in einem festgelegten Zeitintervall nach dem Versenden der Anfrage keine Dokumente ankommen, wird das Empfangsverhalten vom Agenten entfernt. Dieses Behaviour wartet auf Dokumente mit der ConversationID DeliverDocuments und dem Performativ INFORM vom ViewManager. Weiterhin wird in der empfangenen Nachricht ein Ontologiekonzept DeliverDocuments erwartet, das wiederum das Objekt MedicalDocument enthält. Stimmt die Patientenidentifikation der empfangenen Objekte mit dem gesuchten Patienten überein, wird für jedes empfangene MedicalDocument aus dem Ontologie-Objekt DeliverDocuments ein entsprechender Eintrag an der Benutzeroberfläche PersonalAssistantGui erzeugt. Dazu wird in der Methode addDocumentEntry der Klasse PersonalAssistantGui der Oberfläche ein Button hinzugefügt, der die Navigation zu dem assoziierten medizinischen Dokument

realisiert. Mit der Methode addPatientDataEntry der Klasse PersonalAssistantGui wird ein Button an der Oberfläche generiert, der zu den Stammdaten des Patienten navigiert. Die Daten, die für die virtuelle Patientenakte zusammengestellt werden, stammen aus verteilten Informationssystemen. Für jeden Verweis auf der eGK wird eine entsprechende INFORM-Nachricht an den Agenten PersonalAssistantGui gesendet, die, wie eben beschrieben, verarbeitet und visualisiert wird. Diese wiederholte Abarbeitung ist in Abbildung 6.8-12 angedeutet. Das in Abbildung 6.8-13 dargestellte Fenster zeigt die Übersicht über empfangene Dokumente und ihre Strukturierung.

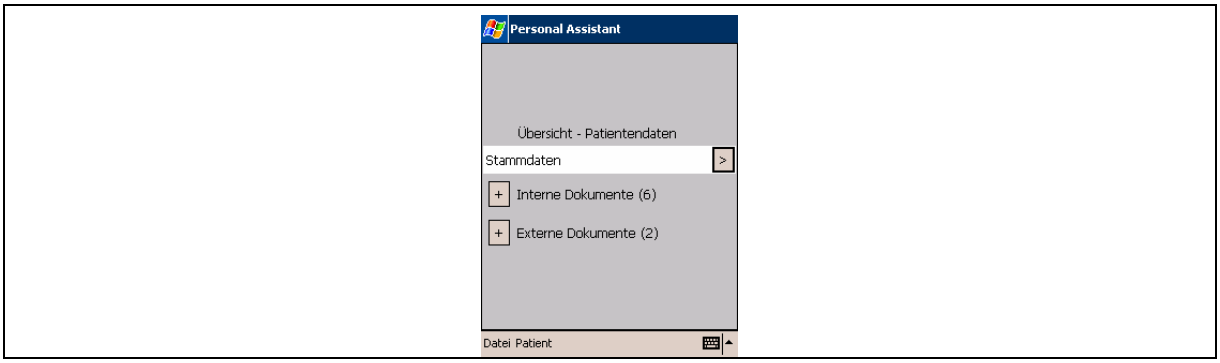

Abbildung 6.8-13: ***Übersichtsfenster zur Darstellung der Navigation zu den aggregierten Dokumenten***
Quelle: Eigene Darstellung

Zum Betrachten der empfangenen Datensätze ist auf den zugehörigen Button zu klicken. Bei Stammdaten wird die Methode showPatientDataPanel der Klasse PersonalAssistantGui aufgerufen. Aus dem dieser Methode übergebenen Ontologie-Objekt Patient werden über die entsprechenden get-Methoden die jeweiligen Werte ausgelesen. Letztere werden in geeignete Elemente der Benutzerschnittstelle zur Anzeige an der Oberfläche eingelesen.

Zur Visualisierung der empfangenen medizinischen Dokumente wird über den entsprechenden Button der Aufruf der Methode showDocument der Klasse PersonalAssistantGui aufgerufen. Das übergebene String-Objekt enthält ein CDA-Dokument in XML-Formatierung. Um dessen Struktur zu parsen, um sie schließlich an der Oberfläche mit geeigneten Elementen der Benutzerschnittstelle anzeigen zu können, wird das Dokument an die Klasse CDAParser übergeben. Die darin implementierten Methoden verwenden Funktionalitäten aus dem XML-Parser Xparse-J (Hillebrand 2006, 81). Aus den Ergebnissen des Parse-Vorgangs werden Überschriften und die jeweils zugehörigen Elemente mit weiterem Inhalt des CDA-Dokuments als AWT-Komponenten dargestellt (siehe Abbildung 6.8-14).

In der in der vorliegenden Arbeit beschriebenen Implementierung der mobilen Anwendung kann die Push-Funktionalität zur Aggregation der medizinischen Dokumente (siehe Abbildung 6.8-15) entweder von der Desktop-Anwendung aus initiiert werden oder proaktiv über die Prozesssteuerung abgerufen werden (siehe dazu Abschnitte 6.7.6 und 6.8.7).

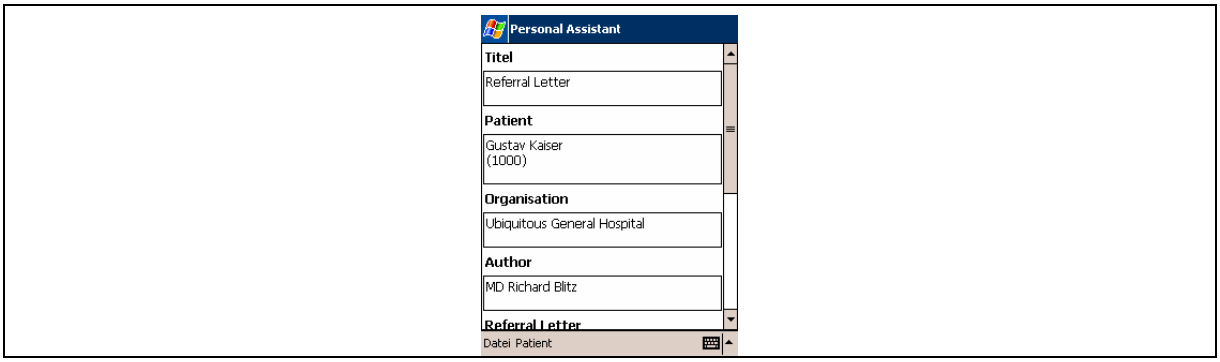

Abbildung 6.8-14: Darstellung eines empfangenen HL7 CDA-Dokuments[81]
Quelle: Eigene Darstellung

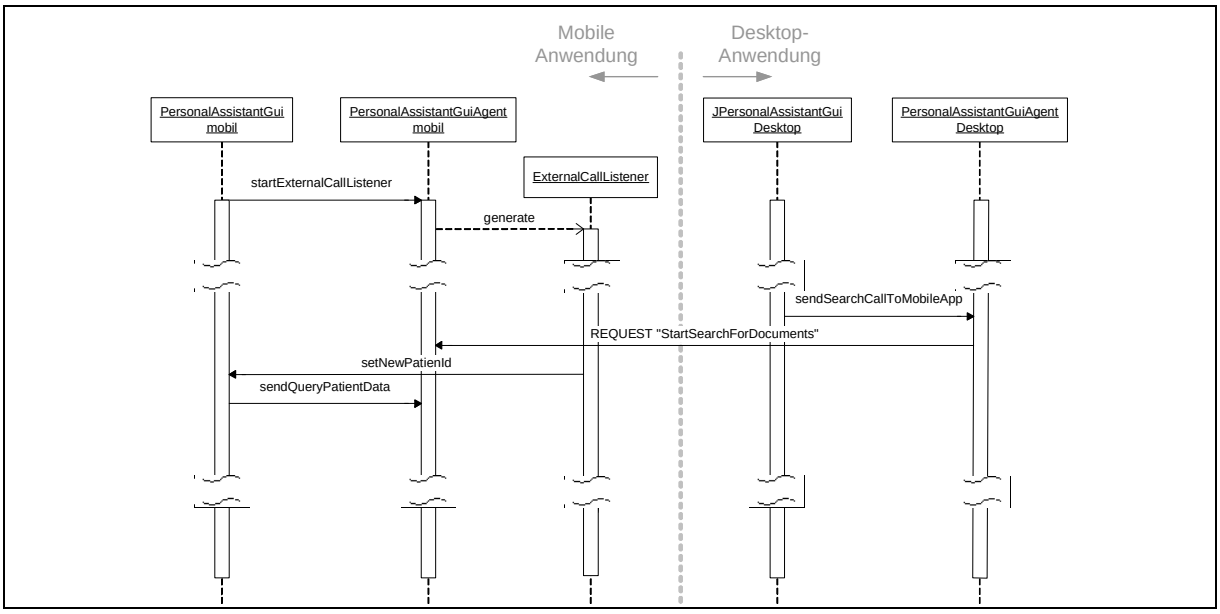

Abbildung 6.8-15: Interaktionen zur Umsetzung der Push-Funktionalität
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

Zur Umsetzung dieser Implementierung wird der Agent PersonalAssistantGuiAgent der mobilen Anwendung in seiner setup-Methode im DF der Umgebung Hospital registriert. In der Methode checkLogin (siehe dazu weiter oben) der Klasse PersonalAssistantGui wird nach dem erfolgreichen Login des Anwenders in der mobilen Anwendung im assoziierten Agenten PersonalAssistantGuiAgent der mobilen Anwendung die Methode startExternalCallListener aufgerufen. In dieser Methode wird das zyklische Behaviour ExternalCallListener in die Warteschlange des Agenten PersonalAssistantGuiAgent zur Verarbeitung eingefügt. Dieses Behaviour verarbeitet eingehende Nachrichten mit der folgenden Spezifikation:

- Performativ REQUEST
- ConversationID StartSearchForDocuments

[81] Zum Inhalt des auf dem Screenshot angezeigten Dokuments siehe die Anmerkung in Fußnote 76. Die persönlichen Daten des gezeigten Patienten sind fiktiv.

In die Desktop-Anwendung ist für den Aufruf der Push-Funktionalität ein eigener Menüeintrag eingefügt. Sobald dieser Eintrag aktiviert wird, wird die Methode sendSearchCallToMobileApp des Agenten PersonalAssistantGuiAgent der stationären Anwendung aufgerufen. In der Methode sendSearchCallToMobileApp wird eine ACL-Nachricht mit den weiter oben spezifizierten Eigenschaften zusammengestellt, der PersonalAssistantGuiAgent der mobilen Anwendung im DF der Umgebung Hospital gesucht und an diesen die soeben konstruierte Nachricht versendet. Wird eine solche Nachricht im PersonalAssistantGuiAgent empfangen, wird über den Aufruf der Methode setNewPatientId der weiter oben beschriebene Mechanismus zur Aggregation der Daten angestoßen. Dazu wird im Agenten PersonalAssistantGuiAgent die Methode sendQueryPatientData aufgerufen. Es folgt die Weiterverarbeitung, wie bereits weiter oben beschrieben.

Mit den Ausführungen in diesem Abschnitt wurde gezeigt, welche Interaktionen zwischen den beteiligten Agenten für die Zusammenstellung der virtuellen Patientenakte im ausgewählten mobilen Endgerät erforderlich sind. Dabei wurden die besonderen Eigenschaften von mobilen Endgeräten berücksichtigt (Anforderung 14 aus Abschnitt 6.4.2), indem die Benutzeroberfläche adäquat angepasst und ein geeigneter Parser für die Verarbeitung der empfangenen HL7 CDA-Dokumente eingesetzt wurde. Somit konnte demonstriert werden, dass sich agentenbasierte Systeme auch auf mobilen Endgeräten einsetzen lassen, um Informationsintegration zu ermöglichen und damit das informationslogistische Prinzip zu unterstützen.

6.8.5 Scheduling

In diesem Abschnitt werden die Funktionalitäten der Scheduling-Anwendung beschrieben. Dazu werden, ausgehend von einer Darstellung in einem Überblick, alle implementierten Verarbeitungsschritte erläutert.

6.8.5.1 Überblick

Beim Starten der gesamten Anwendung werden die in Abschnitt 6.7.4.2 genannten Ressourcen-Agenten für jeweils eine zu kapselnde Ressource initialisiert. Für jede dieser Ressourcen sowie den aktuellen Patienten wird an der Benutzeroberfläche ein Terminplan dargestellt (Abbildung 6.8-16). Die für das Scheduling erforderlichen Patienten-Agenten werden erst dann gestartet, wenn ihre Beteiligung an der Terminverhandlung erforderlich ist, um die Rechnerressourcen erst dann zu allokieren, wenn diese benötigt werden. Die in Abschnitt 6.7.4.2 eingeführten Agenten der Scheduling-Applikation stehen in Relation mit Agenten aus dem bisher beschriebenen System. Somit wird die Scheduling-Applikation nahtlos in die AMD-Anwendung integriert.

Für das Scheduling sind die folgenden Funktionalitäten erforderlich und werden in den Abschnitten 6.8.5.2 mit 6.8.5.9, zusammengefasst aus den Darstellungen bei *Riedl* (2007, 52-66 und 68-71), beschrieben:

- *Visualisierung von Terminplänen*: Zur Anzeige der Termine für den aktuellen Patienten bzw. die Ressourcen werden die im KIS (simuliert) gespeicherten Daten ausgelesen und an der Benutzeroberfläche (siehe Abbildung 6.8-16) angezeigt.
- *Scheduling-Algorithmus*: Durchführung der in Abschnitt 6.7.4.3 beschriebenen Verarbeitungsvorschrift zur Terminplanung

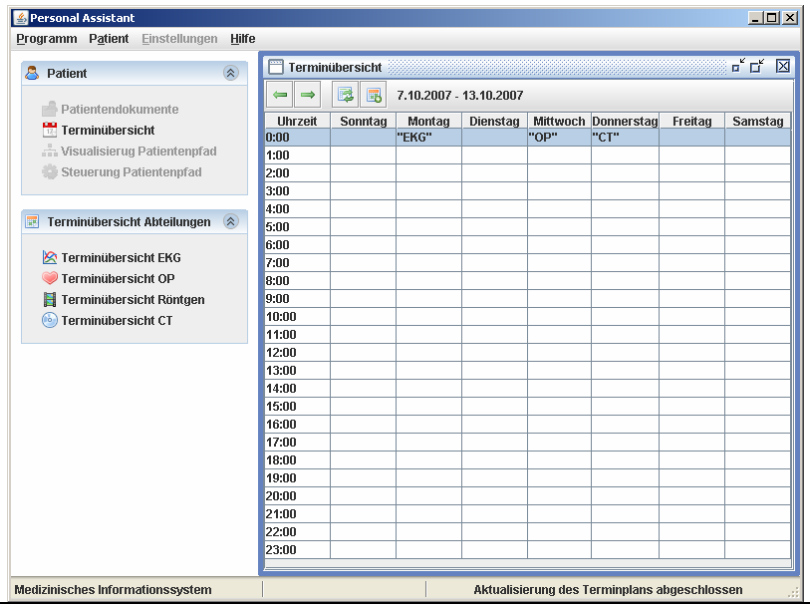

Abbildung 6.8-16: ***Stationäre Anwendung mit Terminübersicht für einen Patienten***
Quelle: Eigene Darstellung

- *Einplanung eines neuen Termins*: Ein neuer Termin für einen Patienten wird in den Kalender eingestellt.
- *Änderung eines bereits vereinbarten Termins*
- *Entfernen eines Termins*
- *Änderung des Gesundheitszustandes für einen Patienten*: Wird der Gesundheitszustand eines Patienten im Sinne des Patienten negativ verändert, wird seine Priorität entsprechend erhöht, womit insgesamt seine Kostenfunktion angepasst wird.
- *Konsistenzsicherung* der Termine: Einhaltung von Reihenfolgerestriktionen zwischen Terminen
- *Berechnung der Startzeitpunkte* von Terminen

6.8.5.2 Visualisierung von Terminplänen

Entsprechend der in Abschnitt 6.8.1 beschriebenen Aggregation von Patientendaten aus dem KIS werden zur Visualisierung der Termine für einen Patienten die analogen Aufrufsequenzen durchgeführt, jedoch mit dem Unterschied, dass kein mediierender Zugriff auf den Kartenleser und die darin eingelegten Karten erfolgt. Dies ist bei der Extraktion von Terminen aus dem lokalen KIS nicht erforderlich; stattdessen wird der Terminplan des Patienten direkt aus dem simulierten KIS gelesen und über die jeweils verantwortlichen Agenten schließlich zur Visualisierung an den PersonalAssistantGuiAgent weitergeleitet. Die Patiententermine sind dabei im lokalen Dateisystem abgelegt und werden über den HISWrapper mit einem simulierten BAPI-Aufruf ausgelesen, weil in der in der vorliegenden Arbeit eingesetzten Version des KISs keine geeigneten BAPIs zur Verfügung stehen. Da die genannten Funktionalitäten bereits in Abschnitt 6.8.1 dargestellt, diese nicht Bestandteil des eigentlichen Scheduling-Algorithmus und für das weitere Verständnis der vorliegenden Arbeit nicht relevant sind, wird in diesem Abschnitt auf weitere Ausführungen dazu verzichtet und stattdessen für eine detaillierte Beschreibung auf *Riedl* (2007, 68) verwiesen.

Die Visualisierung von Ressourcen-Terminplänen erfolgt analog zur Anzeige von Patiententerminen. Die Sequenzdiagramme für die Interaktionen des PatientSchedulingTaskAgents bzw. ResourceSchedulingTaskAgents zur Visualisierung von Terminplänen sind in Anhang A.6.1 bzw. A.6.2 dargestellt.

6.8.5.3 Scheduling-Algorithmus

Die im Folgenden beschriebenen Schritte sind funktional äquivalent mit den in Abschnitt 6.7.4.3 dargestellten Aktionen. Zusätzlich werden in diesem Abschnitt die jeweiligen Behaviours für die Implementierung angegeben (Riedl 2007, 52-56):

1. Die Initiierung eines Scheduling-Durchlaufs wird bei der Änderung eines Patiententermins durch den jeweiligen Patientenagenten ausgelöst.

2. Versand einer Mitteilung vom Patientenagenten (SendInitialRescheduleRequest) an den Ressourcen-Agenten mit dem ursprünglichen und einem potenziellen Alternativtermin sowie Empfang dieser Nachricht vom Ressourcen-Agenten (ReceiveInitialReschedule)

3. Klassifizierung des gewünschten und ursprünglichen Termins des Patienten durch den Ressourcen-Agenten als belegt bzw. frei

4. Versand von Nachrichten vom Ressourcen-Agenten (SendRescheduleConflictInform) an die Patientenagenten, deren Termine sich mit dem gewünschten Termin des Nachfragers überschneiden, sowie Empfang dieser Nachricht bei diesen Patientenagenten (ReceiveRescheduleConflictInform), Versand einer Nachricht vom Ressourcen-Agenten (SendAffectedAgentsCount) an den ursprünglich anfragenden Patientenagenten sowie Empfangen dieser Nachricht (ReceiveAffectedAgentsCount) mit der Anzahl der Patienten, die von der gewünschten Änderung betroffen sind

5. Berechnung der Kosten für die Terminänderung bei den involvierten Patientenagenten, Versand (SendRescheduleCostInform) dieser Kosten an den Initiator des Scheduling-Durchlaufs sowie Empfangen dieser Nachricht beim Initiator (ReceiveScheduleCostInform), Verhandlung der bei der Terminverschiebung involvierten Patientenagenten in der Rolle als Anbieter (SendRescheduleConflict) mit dem Ressourcen-Agenten (ReceiveRescheduleConflict) über einen Alternativtermin

6. Berechnung von Kosten (Methode checkConsistency) und Verschiebung eines Termins, der durch Reihenfolgerestriktionen induziert wird, durch Patientenagenten in der Anbieterrolle, Ausgleich dieser Kosten durch den anfragenden Patientenagenten

7. Berechnung von Kosten bei einer Überschneidung eines Patiententermins bei einer anderen Ressource durch den initiierenden Patientenagenten in der Rolle als Anbieter (Methode checkConsistency)

8. Abarbeitung der Schritte 2 mit 7 durch Patientenagenten in der Rolle eines Anbieters, wenn der von ihnen gewünschte Alternativtermin bereits reserviert ist, andernfalls Mit-

teilung (SendRescheduleCostInform) und Empfang (ReceiveRescheduleCostInform) der ggf. kumulierten Kosten, die durch die Umplanung eines Termins entstehen, vom Anbieter an den Nachfrager, eine rekursive Kostenkumulierung wird so lange fortgeführt, wie ein Nachfrager gleichzeitig in der Rolle als Anbieter fungiert, und terminiert, wenn der ursprüngliche Anbieter dem Initiator des Scheduling-Algorithmus die Kosten für die Terminverschiebungen aller involvierten Patientenagenten mitteilt

9. Entscheidung des initiierenden Agenten, ob mit der angestrebten Terminverschiebung ein Kostenvorteil erreicht werden kann, Versand (SendAcceptRefuseSchedule) der finalen Entscheidung des ursprünglichen Nachfragers an den Ressourcen-Agenten, Empfang (ReceiveAcceptRefuseSchedule) dieser Nachricht im Ressourcen-Agent, Weiterleitung der Änderungen durch den Ressourcen-Agent (SendResourceAcceptRefuseInform) an die betroffenen Patientenagenten (ReceiveAcceptRefuseInform), Festschreibung der Termine entsprechend der Entscheidung des initiierenden Agenten

Diese Darstellung fokussiert den globalen Ablauf der Scheduling-Applikation. Eine detaillierte Beschreibung aus der Sicht der Agenten ResourceSchedulingTaskAgent sowie PatientSchedulingTaskAgent wird in Anhang A.5.13 und A.5.14 dargestellt.

6.8.5.4 Einplanung eines neuen Termins

Wird ein neuer Termin für einen Patienten erstellt (für eine Beschreibung in einem Aktivitätsdiagramm siehe Anhang A.7.1), werden die folgenden Schritte durchlaufen (Riedl 2007, 57-59):

1. Versand einer Nachricht vom AppointmentManager an den PatientSchedulingTaskAgent und Empfangen dieser Nachricht (ReceiveCreateNewAppointment), Einstellen des neuen Termins an das Ende des Patientenplans, um zu gewährleisten, dass keine Konflikte verursacht werden, Versand (SendCreateNewAppointment) dieses Termins durch den Patientenagenten an die Ressource (ReceiveCreateNewAppointment), ggf. Verschiebung dieses Termins bei der Ressource an das Ende des auf dem Zeitstrahl am weitesten hinten liegenden Termins, ggf. Versand dieses Termins von der Ressource (SendNewAppointment) an den Patienten (ReceiveNewAppointment)

2. Einstellen des Termins in das KIS, veranlasst durch den Patientenagenten (SendChangedScheduleInform)

3. Mögliche Einplanung des Patiententermins auf einen früheren Zeitpunkt (ReceiveDeliverSchedule des PatientSchedulingTaskAgents), Beginn des Scheduling-Algorithmus (Abschnitt 6.8.5.3) mit dem potenziell frühesten Zeitpunkt (siehe Abschnitt 6.8.5.9) für den Patienten, wobei hier ggf. Restriktionen durch die Reihenfolge von Terminen zu berücksichtigen sind

4. Kann die Präferenz für den vom PatientSchedulingTaskAgent gewünschten Termin nicht umgesetzt werden, wird der Scheduling-Algorithmus für den nächst späteren, potenziell

frühesten Termin ausgeführt (ReceiveRescheduleAcceptRefuseInform). Dabei werden alle bereits betrachteten Termine als nicht disponierbar klassifiziert.

5. Solange der nächste mögliche Termin vor dem in Schritt 1 identifizierten Termin liegt, wird der Scheduling-Algorithmus iteriert ausgeführt. Sobald ein Termin identifiziert wird, der nach dem in Schritt 1 determinierten Termin liegt, wird auf einen erneuten Durchlauf des Scheduling-Algorithmus verzichtet und der Termin auf das ursprünglich festgeschriebene Zeitintervall, d.h. das Ende des Terminplans des Patienten oder der Ressource, gelegt.

6. Wird über die Durchläufe des Scheduling-Algorithmus ein früherer Termin gefunden, wird dieser als der neue Termin beim Patientenagenten eingetragen (ReceiveReschedule-AcceptRefuseInform).

6.8.5.5 Änderung eines Termins

Änderungen am Terminplan des Patienten können Verlängerungen oder Verkürzungen des vorgesehenen Zeitintervalls sowie Verschiebungen zu einem späteren Zeitpunkt umfassen (Riedl 2007, 59). Für diese Fälle (siehe auch Abbildung A.7-2) werden die folgenden Schritte durchgeführt (Riedl 2007, 59-61):

1. Ist ein Termin entsprechend der Änderung auf der Zeitachse weiter nach hinten zu verschieben, wird derjenige Zeitpunkt berechnet, zu dem der Termin bei dem Patienten am frühesten nach dem ursprünglichen Termin stattfinden kann. Wenn dieser neue Termin später als der gegenwärtige Zeitpunkt ist, wird der ursprüngliche Termin gelöscht und der geänderte Termin als neuer Termin in den Patientenkalender eingefügt. Dazu wird der in Abschnitt 6.8.5.4 beschriebene Algorithmus zur Einplanung eines neuen Termins durchlaufen.
2. Ist die bei 1 beschriebene Bedingung nicht erfüllt, wird überprüft, ob die Dauer des geänderten Termins umfangreicher wurde. Als Konsequenz sind alle folgenden Termine entsprechend auf der Zeitachse nach hinten zu verschieben. Dazu werden entsprechende Nachrichten an die Ressourcen versandt (SendDurationRearrange). Die dafür erforderliche Zeitberechnung ist in Abschnitt 6.8.5.9 beschrieben. Abschließend wird für diesen Fall noch der folgende Punkt 3 überprüft und ggf. abgearbeitet.
3. Kann ein Termin früher als ursprünglich geplant stattfinden, d.h. das potenziell früheste Datum (specifiedDates, siehe Abschnitt 6.8.5.8) wurde auf einen früheren Zeitpunkt festgelegt oder eine durch eine Reihenfolgerestriktion vorgegebene, dem Termin vorgelagerte Untersuchung wurde gelöscht, wird dieser in die Liste collateralRescheduleEarlier (siehe Anhang A.7.1) aufgenommen und schließlich eine potenzielle Einplanung dieses Termins vorgenommen.
4. Sind keine Veränderungen von Terminen erforderlich, werden die geänderten Termine im KIS abgelegt. Diese Bedingung ist einerseits erfüllt, wenn sich die Behandlungsdauer verkürzt. Dabei entstehen zwar potenzielle Verschiebungen von Terminen nach vorne; weil aber keine Konflikte entstehen, wird diese Terminänderung in der aktuellen Implementierung nicht berücksichtigt. Wird andererseits ein durch eine Reihenfolgerestriktion belegter Termin als Nachfolger eines anderen Termins eingetragen, ist der

nachfolgende Termin im Konfliktfall zu aktualisieren, wodurch bereits ggf. entstehende Konflikte aufgelöst werden.

6.8.5.6 Entfernen eines Termins

Das Löschen eines Termins kann Auswirkungen auf nachfolgende Termine besitzen (Riedl 2007, 61). Dazu werden in dem Fall, dass ein Termin die Vorbedingung für andere Termine darstellt, die folgenden Schritte (siehe dazu auch das Aktivitätsdiagramm in Abbildung A.7-3) ausgeführt (Riedl 2007, 61-62):

- Identifikation derjenigen Termine, die zeitlich nach dem gelöschten Termin stattfinden
- Hinzufügen aller dem gelöschten Termin nachgelagerten Termine in die Liste collateralRescheduleEarlier (siehe Anhang A.7.1)
- Entfernen der zeitlichen Abhängigkeit zwischen dem gelöschten Termin und den nachfolgenden Terminen
- Eintragen der so geänderten Termine in das KIS
- Start des Scheduling-Algorithmus für jeden Termin in der Liste collateralRescheduleEarlier, um einen früheren Zeitpunkt für diese Termine zu erhalten

Ist ein zu löschender Termin nicht als Vorbedingung für nachfolgende Termine definiert, kann dieser ohne weitere Nebenbedingungen gelöscht werden (Riedl 2007, 62). Jedoch ergeben sich damit potenzielle Neuverschiebungen für andere Termine. Denn das Löschen eines Termins kann als Spezialfall der Verkürzung der Zeitdauer eines Termins betrachtet werden, indem die Dauer mit dem Wert 0 belegt wird. Diese Funktionalität zur Reduktion der Zeitdauer von Terminen ist in der vorliegenden Implementierung jedoch nicht umgesetzt. Für weitere Ausführungen dazu wird auf Abschnitt 6.8.5.5 verwiesen.

6.8.5.7 Ändern des Gesundheitszustandes für einen Patienten

Verschlechtert sich der Gesundheitszustand des Patienten, erhält er also nach den Erläuterungen in Abschnitt 6.7.4.3 eine höhere Priorität, müssen seine Termine entsprechend angepasst und zeitlich auf dem Zeitstrahl nach vorne verlegt werden (Riedl 2007, 62). Dazu werden die folgenden Schritte (siehe auch Abbildung A.7-4) durchlaufen (Riedl 2007, 62-63):

- Benachrichtigung des Agenten PatientSchedulingTaskAgent über den veränderten Gesundheitszustand
- Hinzufügen der Termine des Patienten zur Liste collateralRescheduleEarlier (siehe Anhang A.7.1)
- Eintragen des geänderten Gesundheitszustandes des Patienten im KIS
- Start des Scheduling-Algorithmus für jeden Eintrag in der Liste collateralRescheduleEarlier

Eine Verbesserung des Gesundheitszustandes von Patienten und die daraus resultierenden Konsequenzen für andere Patientenagenten werden, wie in Abschnitt 6.7.4.3 ausgeführt, nicht weiter betrachtet. Für die vorliegende Implementierung wird deshalb der für den Patienten im

positiven Sinne veränderte Gesundheitszustand nur im KIS persistent abgelegt, ohne Auswirkungen auf einen erneuten Durchlauf des Scheduling-Algorithmus zu haben.

6.8.5.8 Konsistenzsicherung

Zur Gewährleistung der Konsistenz von Terminen werden in der Scheduling-Applikation die folgenden Slots in der im Agentensystem eingesetzten Ontologie definiert (Riedl 2007, 63-64):

- specifiedDates: Angabe eines Datums, vor dem der Termin nicht stattfinden soll. Diese Eigenschaft wird bei der in Abschnitt 6.8.5.9 beschriebenen Berechnung berücksichtigt.
- requiredPreceedingAction: Spezifikation von Terminen, die vor einem bestimmten Termin stattfinden müssen, Berücksichtigung bei der Berechnung in Abschnitt 6.8.5.9
- requiredSucceedingAction: Definition von Terminen, die nach einem bestimmten Termin stattfinden sollen, iterierte Durchführung des Scheduling-Algorithmus, bis keine Inkonsistenzen (Überschneidung von Terminen und erforderliche Reihenfolgerestriktion) durch die Methode checkConsistency festgestellt werden können

Diese Konsistenzsicherung wird beim Durchlaufen der Scheduling-Verarbeitungsvorschrift bzw. insbesondere bei der Berechnung von Startzeitpunkten (siehe Abschnitt 6.8.5.9) vorgenommen (Riedl 2007, 64).

6.8.5.9 Berechung von Startzeitpunkten

Ziel der Berechnung von Startzeitpunkten ist es, im Rahmen von Terminverschiebungen geeignete Alternativen für Termine zu identifizieren. Diese Berechnung in der Methode getNextTimeslot des Agenten PatientSchedulingTaskAgent wird in den folgenden Schritten beschrieben (Riedl 2007, 64-65).

1. Bestimmung des potenziell frühesten Termins für die Behandlung: Addition eines festen Intervalls zum gegenwärtigen Zeitpunkt und lokale Speicherung dieses Termins im Attribut bestTime

2. Ist für den gegebenen Termin ein Datum definiert, für das dieser frühestens eingeplant werden kann (specifiedDates), wird dieses Datum im Attribut bestTime gespeichert, wenn dieses später ist, als das aktuelle Tagesdatum.

3. Wenn der aktuell einzuplanende Termin diesem vorgelagerte Untersuchungen (requiredPreceedingAction) besitzt, wird im Attribut bestTime derjenige Wert gespeichert, der den Endzeitpunkt des letzten vorgelagerten Termins bestimmt. Diese Anpassung erfolgt, wenn bestTime einen Zeitpunkt angibt, der vor den vorgelagerten Untersuchungen ist.

4. Für den aktuellen Termin in bestTime wird sichergestellt, dass keine Überschneidung mit einem anderen Termin vorliegt. Ist dies jedoch der Fall, wird der einzuplanende Termin nach dem Termin verschoben, der die Überscheidung verursacht. Dieser Vorgang wird so lange wiederholt, bis keine Überschneidung mehr identifiziert werden kann.

5. Der lokal gespeicherte Wert in bestTime für den Termin kann als neuer Terminvorschlag in einem weiteren Durchlauf des Scheduling-Algorithmus verwendet werden.

Entsprechend den Ausführungen bei *Riedl* (2007, 65-66) wird für den Fall, dass Termine aufgrund einer längeren als der eigentlich vorgesehenen Zeitdauer auf dem Zeitstrahl nach hinten verschoben werden müssen, vor dem Schritt 1 der Verarbeitungsvorschrift in diesem Abschnitt die folgende Regel eingefügt:

1. Alle Termine, die vor dem verlängerten Termin liegen, werden als belegt klassifiziert. Damit können die Termine auf dem Zeitstrahl nur nach hinten verschoben werden.

Zusammenfassend kann für die Beschreibung der Implementierung der Scheduling-Applikation festgestellt werden, dass Software-Agenten für die Realisierung vorteilhaft eingesetzt werden können. Insbesondere wird damit die Anforderung 10 aus Abschnitt 6.4.1 erfüllt. Außerdem konnte mit diesen Ausführungen demonstriert werden, dass sich agentenbasierte Systeme für die Implementierung eines Informationssystems eignen, das sich in einer von flexiblen Anforderungen geprägten Umgebung befindet.

6.8.6 Patientenportal

Ausgehend von der Beschreibung des Deployments für die Integration des Portals in die Agentenanwendung über WSIG (Anforderung 19 aus Abschnitt 6.4.2) werden im Folgenden die Schritte bei der Initialisierung der Applikation sowie der Visualisierung der virtuellen Patientenakte im Portal beschrieben:

Die Architektur für das Patientenportal einschließlich der Agentenanwendung ist in Abbildung 6.8-17 dargestellt und wird im Folgenden erläutert: Apache jUDDI[82] (für Ausführungen zu UDDI siehe Abschnitt 3.2.4.4) läuft als Dienstverzeichnis in einem eigenen Applikationsserver (Apache Tomcat[83]). Über diesen Applikationsserver wird die Funktionalität von jUDDI anderen Aufrufern zur Verfügung gestellt. Weiterhin wird ein Applikationsserver mit einem Container für das gewählte Portalsystem Liferay eingesetzt. Als Container fungiert dabei JBoss[84], welcher die Rolle des EJB-Containers für Liferay einnimmt. Im Datenbankserver, im vorliegenden Fall ein MySQL®[85]-Server, werden die Daten für jUDDI[86] und Liferay persistent abgelegt. Auf diese Datenbank wird über HTTP/SOAP lesend und schreibend zugegriffen. Über ihre Applikationsserver können jUDDI und Liferay (Anforderung 17 aus Abschnitt 6.4.2) mit HTTP/SOAP auf ihre Daten in den jeweiligen Datenbanktabellen zugreifen. Die Kommunikation zwischen WSIG und den beiden Applikationsservern basiert ebenfalls auf HTTP/SOAP.

[82] Informationen zu jUDDI können auf den folgenden Seiten abgerufen werden: http://ws.apache.org/juddi/, zugegriffen am 24.01.2007.
[83] Siehe dazu http://tomcat.apache.org/, zugegriffen am 24.01.2007.
[84] Für weitere Informationen siehe http://www.jboss.com/, zugegriffen am 24.01.2007.
[85] Siehe dazu die folgenden Seiten: http://www.mysql.com/, zugegriffen am 24.01.2007.
[86] Da Liferay auf der Datenbank MySQL basiert, wird auch für die Speicherung der jUDDI-Daten diese Datenbank gewählt. Für diesen Zweck könnte aber auch eine andere relationale Datenbank gewählt werden.

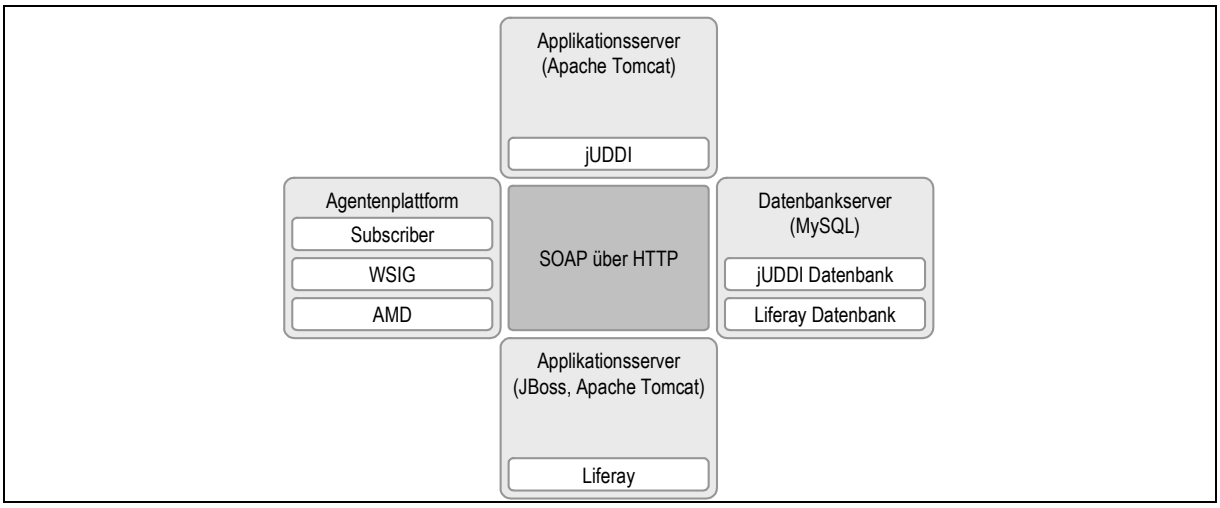

Abbildung 6.8-17: Architektur für das Patientenportal
Quelle: Eigene Darstellung, in Anlehnung an *Druker/Michelbach* (2006, 55)

Zum Starten der gesamten Anwendung werden die folgenden, für die Integration des Portals in das MAS relevanten Schritte durchgeführt (siehe dazu auch Druker/Michelbach 2006, 55-56): Bei der Initialisierung des WSIG-GatewayAgent erfolgt seine Registrierung im DF der Umgebung Hospital[87]. Dieser Agent führt die in Abschnitt 6.3.5 beschriebenen Aufgaben durch und fungiert somit als Schnittstelle zwischen der Web Services- und der Agentenapplikation. Um die Web Services-Funktionalität geeignet in die bisherige Architektur des AMD zu integrieren, wird beim Öffnen der virtuellen Patientenakte ein spezieller Subscriber-Agent (siehe dazu auch Druker/Michelbach 2006, 53-54 und 56-57) initialisiert und am DF der Umgebung Hospital registriert. Mit diesem Architektur-Ansatz wird der Subscriber-Agent zu einer Komponente im modularen AMD (siehe dazu die Darstellung in Abschnitt 6.5.1).

Der vereinfachte Kommunikationsmechanismus (siehe Abbildung 6.8-18) zwischen dem Patientenportal und der Agentenanwendung (Anforderung 17 aus Abschnitt 6.4.2) wird im Folgenden beschrieben: Wird vom Patienten die Einsichtnahme in die virtuelle Patientenakte von der Benutzerschnittstelle des Portals angefordert, wird eine Web Services-Nachricht versendet und über den Axis Web Server an den GatewayAgent geeignet weitergeleitet. Letzterer identifiziert die für den angeforderten Dienst erforderliche Schnittstelle aus dem jUDDI-Verzeichnis. Mit dieser Information kann vom GatewayAgent eine entsprechende ACL-Nachricht an den Subscriber gesendet werden. Dort erfolgt die Interaktion[88] mit der Agentenapplikation zur Aggregation von Dokumenten:

[87] Weil das gewählte Szenario in einem Krankenhaus angesiedelt ist, wird dieses auch für die Registrierung der WSIG-Dienste eingesetzt.

[88] Zur Reduktion der Komplexität wurde in der prototypischen Implementierung zur Ankopplung des Portals an die Agentenanwendung angenommen, dass die Patientenakte in der Umgebung Hospital bereits geöffnet ist. Diese Einschränkung ist gerechtfertigt, weil das Ziel die Fokussierung auf die Integration des Patientenportals in die Agentenanwendung darstellt. Somit sind keine Einschränkungen hinsichtlich der Qualität der Ergebnisse gegeben, da die vollständigen Interaktionsmuster im Szenario für die Dokumentenaggregation bereits in Abschnitt 6.8.1 beschrieben wurden.

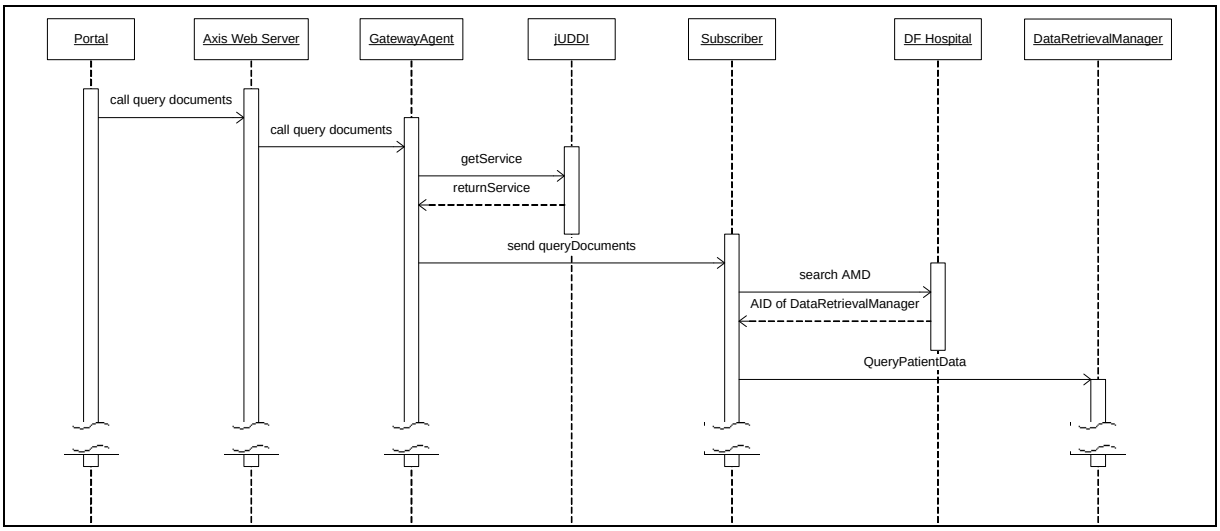

Abbildung 6.8-18: ***Kommunikation zwischen dem Patientenportal und dem AMD***
Quelle: Eigene Darstellung, in Anlehnung an *Druker/Michelbach* (2006, 57)

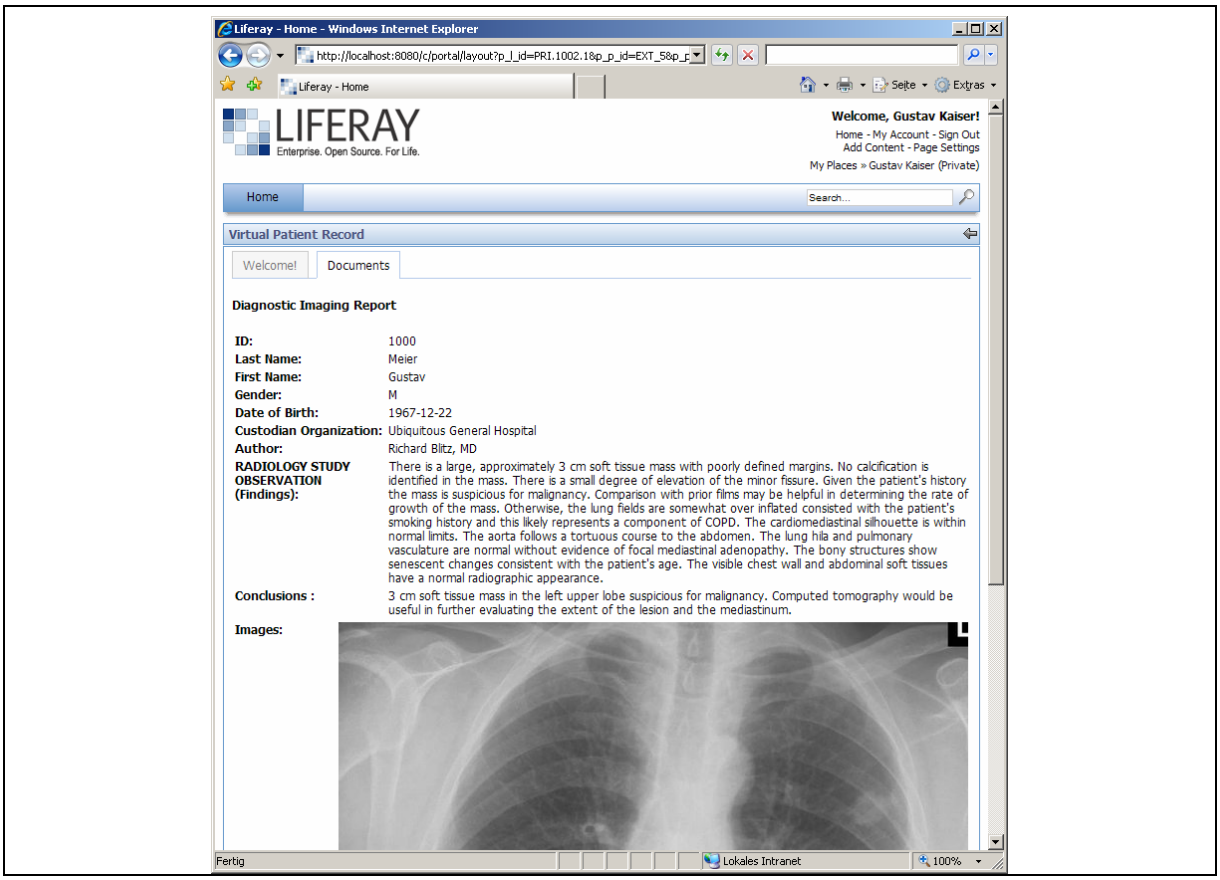

Abbildung 6.8-19: ***Visualisierung eines Dokuments aus der virtuellen Patientenakte im Portal***
Quelle[89]: Eigene Darstellung

Sobald die Adresse des DataRetrievalManager in der Rolle des zentralen Einstiegspunkts in das AMD aus der Umgebung Hospital nach dem Suchvorgang (search AMD) vorliegt, kann die

[89] Zur Herkunft der Daten und des Bildmaterials siehe die Anmerkungen zu den Fußnoten 76 und 77.

Aggregation der Dokumente der virtuellen Patientenakte über die Kommunikation mit dem DataRetrievalManager angestoßen werden. Dort wird diese Nachricht entsprechend den Ausführungen in Abschnitt 6.8.1 in der Agentenplattform weiterverarbeitet.

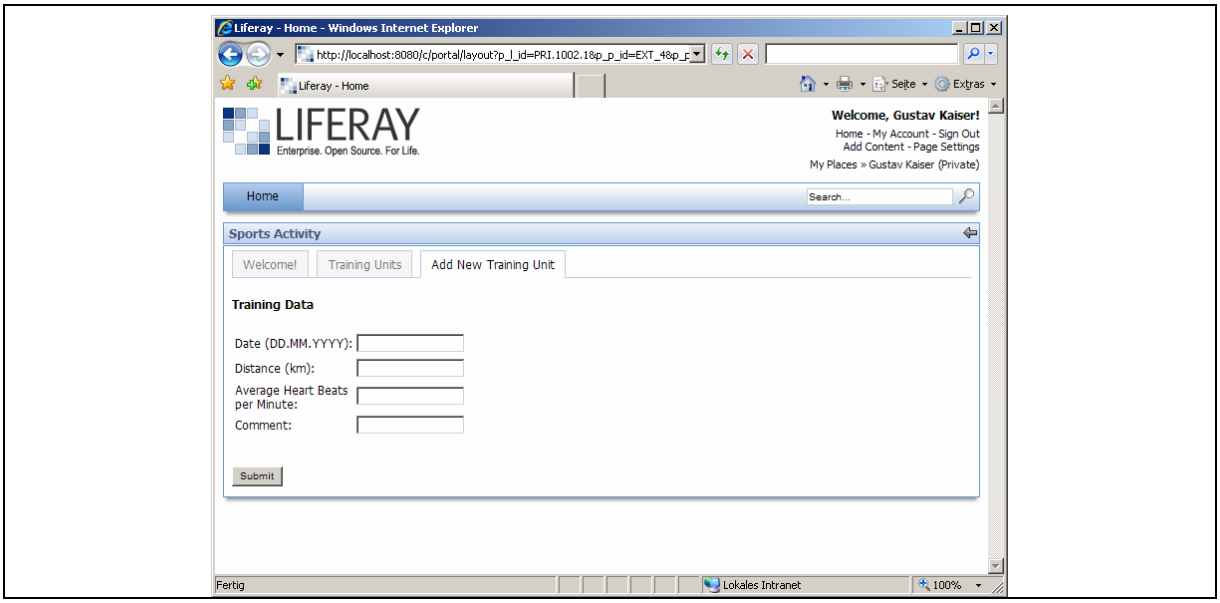

Abbildung 6.8-20: ***Eingabe von Daten über die sportliche Aktivität des Patienten***
Quelle: Eigene Darstellung

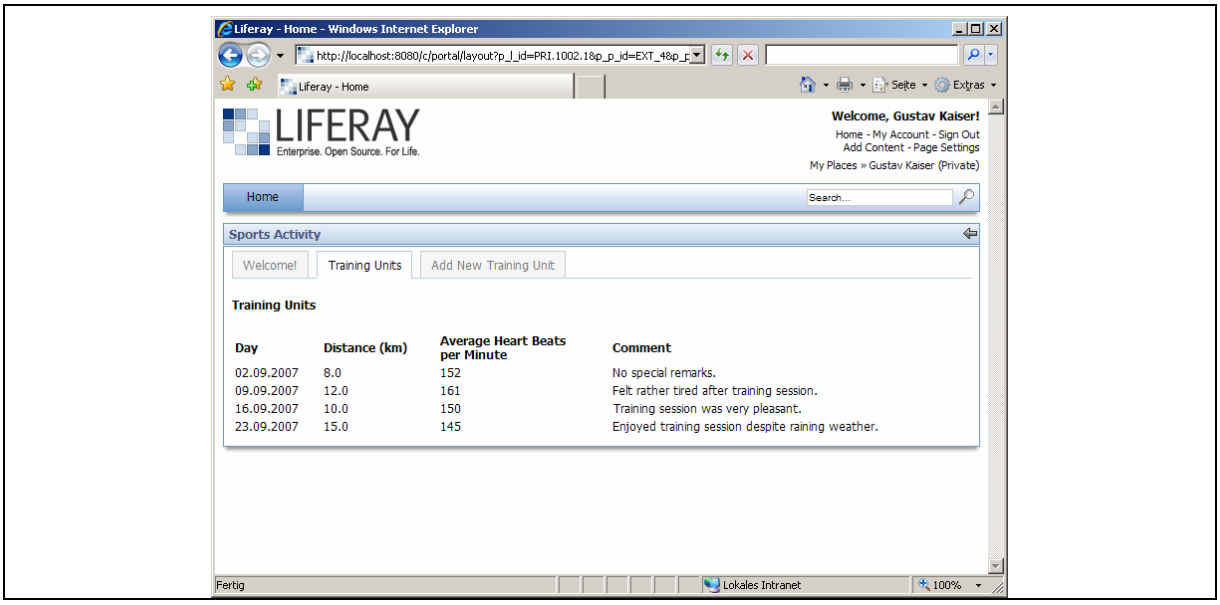

Abbildung 6.8-21: ***Visualisierung von Daten über die sportliche Aktivität des Patienten***
Quelle: Eigene Darstellung

Die aus dieser Abfrage resultierenden Dokumente werden über den Subscriber (Behaviour ReceiveDeliverDocuments), den GatewayAgent und den Axis Web Server schließlich an das Portal weitergeleitet. Dazu werden die Nachrichten geeignet in das SOAP-Format transformiert. Schließlich werden die empfangenen Daten für die Visualisierung an der Benutzeroberfläche über eine XSLT-Definition transformiert und angezeigt (siehe Abbildung 6.8-19).

Neben der weiter oben beschriebenen Anzeige von Daten aus der virtuellen Patientenakte erlaubt das Patientenportal auch die Eingabe (siehe Abbildung 6.8-20) und Visualisierung (siehe Abbildung 6.8-21) von Daten über die sportlichen Aktivitäten des Patienten. Damit wird die gewünschte Integration des Patienten in den Behandlungsprozess erreicht.

Zusammenfassend lässt sich feststellen, dass eine Ankopplung eines Patientenportals an das MAS technisch gelingt. In der Implementierung für die vorliegende Arbeit werden dazu Daten der virtuellen Patientenakte visualisiert. Außerdem erhält der Patient die Möglichkeit, eigene Daten einzugeben und einzusehen. Diese Funktionalität ist jedoch so gestaltet, dass die Interpretation der eingegebenen Daten durch den Leistungserbringer den Einfluss auf die Entscheidungsfindung möglichst gering hält. Auf diesem Weg wird der in Abschnitt 4.9.2 geforderte Kompromiss zwischen der Patientenintegration in den Behandlungsprozess und der Qualitätssicherung der in der virtuellen Patientenakte aggregierten Daten erreicht. Somit können insgesamt die Anforderungen 11 aus Abschnitt 6.4.1 sowie 17 aus Abschnitt6.4.2 erfüllt werden.

6.8.7 Agentenbasierte Prozessunterstützung

Die Prozessunterstützung wird entsprechend der in Abschnitt 6.7.6.2 dargestellten Architektur in zwei Phasen durchlaufen: Einlesen und persistentes Ablegen der Prozessinformation (Phase 1, siehe dazu auch Abbildung 6.7-9) sowie Ausführen des ausgewählten Prozesses (Phase 2, siehe Abbildung 6.7-11). Die dafür erforderliche Implementierung wird, zusammengefasst aus den Darstellungen bei *Bergmann/Klinger/Winkler* (2006, 27-73), in den Abschnitten 6.8.7.1 mit 6.8.7.3 dargestellt.

6.8.7.1 Überblick

In diesem Abschnitt wird ein Überblick über die Schritte der Phasen 1 und 2 sowie die dafür jeweils erforderliche Implementierung beschrieben. Phase 1 umfasst die folgenden Schritte, wobei die jeweiligen Pakete der Implementierung und ihre Bedeutungen in Tabelle 6.8-5 zusammengefasst sind. Dabei ist zu beachten, dass nur diejenigen Klassen, Pakete und ihre jeweiligen Funktionalitäten berücksichtigt werden, die für die Prozesssteuerung erforderlich sind.

- Einlesen der Prozessinformationen (Paket processRepository)
- Parsen der AML-Datei (processRepository.aris) und Generieren des EPK-Modells (processRepository.epk.model)
    - Generieren der Objektstruktur für die eingelesenen AML-Dateien
    - Hinzufügen der Referenzen zwischen den Knoten
    - Auflösen der Referenzen der Prozesswegweiser durch Identifikation der Einstiegs- und Rückkehrpunkte
- Validierung der eingelesenen EPK (processRepository.epk.validation)
- Transformation des EPK-Modells in eine PML-Objektrepräsentation (processRepository.processDef)
    - Mapping einer EPK-Funktion zu einem im EPKSubProcessMapping bestimmten Objekt
    - Identifikation der Nachfolger in der EPK-Darstellung und Setzen dieser als Vorbedingungen in PML

- Löschen eines ggf. vorhandenen, gleichnamigen EPK- und PML-Modells über Hibernate aus einer relationalen Datenbank, Speichern der Modelle über Hibernate in einer relationalen Datenbank
- Generierung der PML-Datei[90] aus der PML-Objektrepräsentation und Ablegen im lokalen Dateisystem

Paket	Kurzbeschreibung
processRepository	• Objektrepräsentation der XML-Datei processImport.xml und ihrer Elemente • Beschreibung der zu ladenden Teilprozesse, die einen vollständigen Behandlungsprozess determinieren • Lesen und Schreiben von persistenten Objekten im Dateisystem oder in der Datenbank
processRepository.aris	• Parsen der AML-Beschreibung • Objektrepräsentation für eine AML-Beschreibung • Darstellung der Datenstruktur durch Speicherung der Elemente und ihrer jeweiligen Vorgänger
processRepository.epk	• Objektrepräsentation der Schemadatei epkProcessMapping.xsd • Transformation einer EPK in das Format PML
processRepository.epk.exception	• Beschreibung einer Ausnahme beim Auftreten einer reflexiven Relation eines Operators oder Prozesswegweisers in der EPK-Struktur • Beschreibung einer Ausnahme, wenn Referenzen durch einen Prozesswegweiser innerhalb einer EPK nicht bestimmt werden können
processRepository.epk.model	• Objektrepräsentation einer EPK • Definition von abstrakten Methoden zur Validierung einer EPK und zur Transformation zu PML • Spezifikation der Hibernate-Abbildung über Annotationen im Programmtext
processRepository.epk.validation	Validierung einer eingelesenen EPK
processRepository.processDef	• Objektrepräsentation eines Prozesses im PML-Format • Darstellung der Datenstruktur über die Speicherung der Elemente und ihrer jeweiligen Vorbedingungen • Spezifikation der Hibernate-Abbildung über Annotationen im Programmtext

Tabelle 6.8-5: ***Paketstruktur zur Verarbeitung und Transformation der Prozessdateien***
Quelle: Eigene Darstellung, Zusammenfassung der Ausführungen bei *Bergmann/Klinger/Winkler* (2006, 27-38)

In Phase 2 der Prozessunterstützung wird die Prozessinformation aus einer relationalen Datenbank ausgelesen und für die Prozesssteuerung eingesetzt. Dazu werden die folgenden Schritte ausgeführt (Bergmann/Klinger/Winkler 2006, 53):

- Auswahl eines Behandlungspfades im PML-Format über die Benutzeroberfläche
- Einlesen des gewählten Prozesses
- Interpretation der Prozessbeschreibungssprache PML und weitere Transformation dieser Informationen zur Steuerung des gewählten Prozesses (ProcessorFactory)
- Aktivierung des Prozesses über den Start-Button der Prozesssteuerung

[90] Diese Datei ist erforderlich, weil die Prozesssteuerung eine solche PML-Datei als Eingabe zur Verarbeitung erwartet.

- Steuerung des Prozesses durch externe Ereignisse (Benutzereingaben und Interaktion mit dem Simulationssystem SeSAm, siehe dazu Abschnitt 6.8.8)
- Visualisierung der Prozessschritte an der Benutzeroberfläche

Ein Überblick über zentrale Klassen für die Implementierung der Prozesssteuerung und ihre jeweilige Funktionalität ist in Tabelle 6.8-6 dargestellt.

Klasse	Kurzbeschreibung
ProcessorFactory	Parsen und Transformation von PML
ProcessMasterAgent	• Einlesen des gewählten Prozesses • Steuerung des Prozesses durch das Behaviour ProcessorBehaviour
KillAgent	Beenden des ProcessMasterAgent
ProcessorBehaviour	• Implementierung der Schnittstelle IProcessingBehaviour • Scheduling der Prozessschritte • Überprüfung der Vorbedingungen zur Auswahl für den nächsten Schritt
ProcessorState	Speicherung der Attribute aus ProcessorBehaviour, die den Zustand (Fortschritt) des aktuellen Prozesses beschreiben
ProcessState	• Speicherung von relevanten Attributen zum persistenten Ablegen und Wiederaufnahmen eines Prozesses • Definition einer Hibernate-Abbildung über Annotationen im Programmtext
IProcessPersister	Schnittstelle zum Lesen, Suchen sowie Speichern von ProcessState-Objekten in einer Datenbank über Hibernate
ProcessPersister	Implementierung der Schnittstelle IProcessPersister
Repository	• Speicherung der Ergebnisse von Prozessen • Implementierung nach dem Composite-Entwurfsmuster (Gamma et al. 1995, 163-173)
ConstraintSet	• Kapselung einer Vorbedingung als Repository-Objekt (siehe dazu die Darstellung in Abschnitt 6.7.6.1) • Vergleich einer Vorbedingung mit den Ergebnissen eines Prozesses • Implementierung nach dem Composite-Entwurfsmuster (Gamma et al. 1995, 163-173) • Unterstützung der Konnektoren AND, OR und XOR (entsprechend der Funktionalität einer EPK)

Tabelle 6.8-6: ***Paket- bzw. Klassenstruktur zur Prozesssteuerung***
Quelle: Eigene Darstellung, Zusammenfassung der Ausführungen bei *Bergmann/Klinger/Winkler* (2006, 53-58)

Die für die Implementierung der Benutzeroberfläche verwendeten Klassen sind in Tabelle 6.8-7 zusammengefasst. Dabei implementieren die Klassen ProcessorTreeModel (Modell), ProcessorIntFrame (View) und CompositeProcessorController (Controller) das MVC-Prinzip (Krasner/Pope 1988), welches in Abschnitt 3.2.3.1 beschrieben wird.

Klasse	Kurzbeschreibung
CompositeProcessorController	• Implementierung der Schnittstelle IProcessorController nach dem MVC-Muster (Krasner/Pope 1988) • Referenziert UIProcessorController sowie GraphUIProcessorController und leitet Methodenaufrufe an diese Controller-Klassen weiter
GraphUIProcessorController	Schnittstelle für die grafische Darstellung des aktuellen Prozesses als EPK
UIProcessorController	Steuerung der Benutzeroberfläche über die Interpretation der Informationen zum Verlauf der Prozessverarbeitung
ProcessorIntFrame	• Visualisierung des aktuellen Prozesszustandes als Baum bzw. Text • Bedienung der Prozesssteuerung • View-Implementierung nach dem MVC-Muster (Krasner/Pope 1988)

Tabelle 6.8-7: ***Paket- bzw. Klassenstruktur für die Benutzeroberfläche***
Quelle: Eigene Darstellung, in Anlehnung an *Bergmann/Klinger/Winkler* (2006, 56-58)

6.8.7.2 Transformation

Das Einlesen der AML-Prozessinformation, ihre Transformation sowie das persistente Ablegen in einer Datenbank bzw. im Dateisystem werden in den folgenden Schritten (Abbildung A.8-3) ausgeführt und in den weiteren Darstellungen detailliert beschrieben (Bergmann/Klinger/Winkler 2006, 38-52):

- Einlesen der Prozessinformation für einen Behandlungspfad
- Parsen der AML-Beschreibung
- Dereferenzieren von Prozesswegweisern
- Validierung
- Transformation zu PML
- Persistentes Ablegen der AML- und PML-Beschreibung

Bevor diese Schritte beschrieben werden, wird der gesamte Prozess der Transformation dargestellt (Abbildung A.8-3): In der Methode main wird eine Konfigurationsdatei eingelesen (Methode loadImports der Klasse ProcessImport, Anforderung 15 aus Abschnitt 6.4.2), welche die Dateien mit den eigentlichen Prozessinformationen spezifiziert. Das Ergebnis dieses Methodenaufrufs wird dem Konstruktor für das Objekt vom Typ ARISParser übergeben, auf dem die Methode parse zum Erstellen der EPK-Objekrepräsentation entsprechend den eingelesenen Prozessinformationen aufgerufen wird. Mit der Methode getModel wird diese Objektdarstellung für die Validierung über das Objekt EPKValidator zur Verfügung gestellt. Die eigentliche Validierung wird durch die Methode logAllErrors initiiert, in der die Methode validate aufgerufen wird. Wenn bei der Validierung keine Fehler identifiziert werden, wird die EPK-Objektstruktur vom Typ EPKModel durch den Aufruf der Methode transformToProcessXML zu einer PML-Objekstruktur transformiert. Die EPK- und PML-Objektstruktur werden in der Methode replaceEPKModel bzw. replaceProcessDef weiterverarbeitet: Wenn bereits Modelle mit jeweils dem gleichen Namen in der Datenbank existieren, werden diese zunächst über delete gelöscht. Anschließend werden die Objektstrukturen über save in der Datenbank persistent ab-

gelegt. Abschließend wird in der Methode saveProcessDef die PML-Objektstruktur ProcessDef im Dateisystem gespeichert. Im Folgenden werden die im Überblick dargestellten Verarbeitungsschritte der Transformation detailliert beschrieben:

Da ein Prozess in der AML-Beschreibung aus mehreren Teilprozessen bestehen kann, sind zunächst Informationen über diese zu integrierenden Teilprozesse zu importieren. Dazu wird in der Datei processImport.xml für den gewählten Prozess spezifiziert, welche AML-Dateien, die die Teilprozesse des Gesamtprozesses repräsentieren, und welche zugehörigen ProcessMappingXML-Dateien (subProcesses_<name>.xml) einzulesen und für die weitere Transformation bereitzustellen sind. Die eigentliche Prozessinformation wird über die Methode parse im AML-Format eingelesen und als Objektstruktur dargestellt. Nach erfolgreicher Validierung der eingelesenen Prozessinformation wird die EPK-Objektstruktur in die PML-Struktur transformiert. Die AML- und PML-Objektstrukturen werden abschließend in der relationalen Datenbank persistent abgelegt. Die PML-Objektstruktur wird zusätzlich im Dateisystem gespeichert, um als Eingabe für die spätere Prozesssteuerung fungieren zu können.

Für den Parse-Vorgang der AML-Beschreibung ist relevant, dass das AML-Format und die EPK-Objektstruktur jeweils unterschiedliche Datenstrukturen besitzen. In AML referenzieren Knoten jeweils ihre Nachfolger, bei der EPK-Objektstruktur wird für einen Knoten jeweils der Vorgänger angegeben. Damit ist eine geeignete Transformation der Datenstrukturen erforderlich. Diese Transformation ist deshalb nötig, weil die Prozesssteuerung auf der Basis von erfüllten Vorbedingungen die durchzuführenden Prozessschritte auswählt. Dafür ist jeweils die Information über die Vorgänger erforderlich. Der gesamte Vorgang zum Parsen der AML-Struktur ist in Abbildung A.8-4 dargestellt. Das Parsen der AML-Dateien stützt sich auf das Element OjbOcc. Für jeden Teilprozess (EPK-Prozesswegweiser) im Gesamtprozess wird eine Instanz vom Typ ARISSubDoc erstellt, durch welche jeweils die Objektstruktur der AML-Beschreibung abgebildet wird. Für jeden erstellten Teilprozess wird die Methode createSubModel aufgerufen. In Abhängigkeit der gefundenen AML-Elemente werden entsprechende EPK-Objekte generiert. Um die Relationen zwischen den Objekten zu erhalten, wird die AML-Struktur erneut betrachtet und über die Methode addPredecessor werden entsprechende Vorgängerinformationen hinzugefügt.

Die Schritte zum Dereferenzieren der Prozesswegweiser (Methode resolveSubProcesses) sind in Abbildung A.8-5 dargestellt und werden im Folgenden beschrieben: Ist ein Prozesswegweiser sowohl Einstiegspunkt als auch Rückkehrpunkt für einen referenzierten Teilprozess, wird ein zusätzlicher Prozesswegweiser in den referenzierenden Teilprozess integriert (Methoden split und addNode), um die Vorbedingungen für die spätere Prozessausführung im referenzierenden Teilprozess eindeutig bestimmen zu können. Weiterhin werden über die Methoden setEndLinkDummy und setStartLinkDummy Verweise zwischen dem ursprünglichen und generierten Prozesswegweiser hergestellt. Damit wird gekennzeichnet, dass ein Prozesswegweiser sowohl Einstiegs- als auch Rückkehrpunkt ist.

Im weiteren Verlauf der Dereferenzierung werden für die referenzierenden Prozesswegweiser die Einsprungpunkte in die zugehörigen Teilprozesse (Methode findMatchingSubProcess) über die Identifikation durch die Bezeichnung der Teilprozesse gesucht. Falls diese Suche erfolglos ist, wird nach Ereignissen ohne Vorgänger (Methode findStartEvents) gesucht, die die gleiche

Bezeichnung wie dasjenige Ereignis besitzen, das dem referenzierenden Prozesswegweiser unmittelbar vorausgeht. Damit können die Referenzierungen als Einsprungpunkte zwischen Prozesswegweisern und Teilprozessen ggf. aufgelöst werden. Können diese Einsprungpunkte erfolgreich aufgelöst werden, sind die Rückkehrpunkte zu determinieren. Dafür werden Prozesswegweiser gesucht, die die gleiche Bezeichnung wie der referenzierende Teilprozess und keinen Nachfolger besitzen. Wird die Referenzierung damit nicht aufgelöst, wird nach Ereignissen gesucht, die keine Nachfolger haben und demjenigen entsprechen, das dem Prozesswegweiser des referenzierenden Teilprozesses unmittelbar folgt.

Nach der erfolgreichen Identifikation von Einsprung- und Rückkehrpunkten werden die Verweise zwischen den zusammengehörenden Teilprozessen gesetzt: Der Einsprungpunkt im referenzierten Teilprozess erhält (Methode addPredecessor) alle Vorgänger des referenzierenden Prozesswegweisers. Besitzt ein Prozesswegweiser die Doppelrolle eines Einsprung- und Rückkehrpunktes, erhält (Methode addLinkedEndNodes) der generierte Prozesswegweiser Referenzen auf alle Vorgänger des referenzierten Teilprozesses.

Die wie beschrieben generierte EPK-Objektstruktur wird abschließend auf ihre syntaktische Korrektheit validiert. Für einzelne Knoten wird dazu in den Klassen EPKEvent, EPKOperator, EPKProcess (EPK-Funktion) und EPKSubProcess (EPK-Prozesswegweiser) jeweils die Methode validate aufgerufen. Die syntaktische Korrektheit der gesamten EPK wird über den Aufruf der Methode validate der Klasse EPKValidator überprüft. Die Validierung orientiert sich an den EPK-Syntaxregeln (für eine Darstellung siehe Tabelle A.9-1 und Tabelle A.9-2). Bei der Validierung werden insbesondere die Fehler berücksichtigt, bei deren Auftreten eine Weiterverarbeitung nicht möglich ist:

Beim Auftreten von reflexiven Relationen bei Operatoren und Prozesswegweisern in einer EPK würde die weiter unten beschriebene Transformation der AML- zur PML-Objektstruktur nicht terminieren. Deshalb wird bei der Validierung überprüft, ob solche Eigenschaften vorliegen. Weiterhin wird gewährleistet, dass in einem Prozess ein Startereignis, mindestens eine Funktion oder ein Prozesswegweiser mit Vorgänger identifiziert werden können.

Bezüglich Warnungen bei der Validierung, deren Auftreten nicht die Beendigung der Weiterverarbeitung impliziert, ist insbesondere zu erwähnen, dass die Überprüfung auf bipartite Alternierung von Funktionen und Ereignissen aktiviert oder deaktiviert werden kann. Somit wird die Möglichkeit gegeben, komplexe Behandlungspfade aus Gründen der Übersichtlichkeit ohne bipartite Alternierung darzustellen und in der Prozesssteuerung zu verarbeiten.

Zur Transformation (Methode transformToProcessXML in Abbildung A.8-6) der so konstruierten EPK-Objektstruktur zur PML-Objektstruktur werden die folgenden Schritte abgearbeitet: Für den Wurzelprozess wird über die Methode generateMasterProcess ein Objekt vom Typ ProcessDef erzeugt. Für jeden durch einen Prozesswegweiser referenzierten Teilprozess (EPKSubModel) werden alle EPK-Funktionen (EPKProcess) betrachtet. Um diese EPKProcess-Objekte auf ProcessDef-Objekten abzubilden, wird über die Methode generateProcessDef das über EPKProcessMappingXML spezifizierte ProcessDef-Objekt identifiziert. Die Vorbedingungen für dieses ProcessDef-Objekt werden durch die Methode generateConstraints bestimmt: Über die darin aufgerufene Methode generateFinishedConstraints werden die Bedingungen des

referenzierten Vorgängerknotens definiert, die erfüllt sein müssen, bevor der folgende Prozessschritt ausgeführt werden kann. Dabei werden die in Tabelle 6.8-8 beschriebenen Vorschriften in Abhängigkeit des Knotentyps von der Methode generateFinishedConstraints abgearbeitet.

Knotentyp	Verarbeitung der Methode generateFinishedConstraints
EPKEvent (EPK-Ereignis)	Sind im EPKProcessMappingXML Constraints für das jeweilige EPKEvent angegeben, werden nur diese zurückgegeben. Einen Sonderfall bilden Startereignisse. Diese geben nur die Standard-Constraints „::process_started“ zurück, welche beim Prozessstart automatisch vom System generiert werden. Trifft keiner der genannten Fälle zu, werden die erfüllten Bedingungen (rekursiver Aufruf von generateFinishedConstraints) des Vorgängers zurückgegeben.
EPKProcess (EPK-Funktion)	Gibt die Constraints zurück, die erfüllt sind, wenn dieser EPKProcess beendet ist.
EPKOperator (EPK-Operator)	Gibt eine ConstraintCollection mit dem Typ des Operators („AND“, „OR“ oder „XOR“) zurück, die die erfüllten Bedingungen (rekursiver Aufruf von generateFinishedConstraints) aller Vorgänger verknüpft.
EPKSubProcess (EPK-Prozesswegweiser)	Erbt das Verhalten von der Klasse EPKOperator.

Tabelle 6.8-8: ***Verarbeitung der Methode generateFinishedConstraints in Abhängigkeit des jeweiligen Knotentyps***
Quelle: *Bergmann/Klinger/Winkler* (2006, 46)

Über den Aufruf der Methode setConstraints werden die gefundenen Vorbedingungen im zugehörigen ProcessDef-Objekt belegt. Diese werden mit ggf. bereits existierenden Vorbedingungen im ProcessDef-Objekt zusammengefasst. Der nun vollständig definierte Prozessschritt wird abschließend über die Methode addProcessDef Teil des Wurzelprozesses.

Für das persistente Ablegen der AML- und PML-Objekstruktur in einer Datenbank werden im Programmtext Annotationen eingefügt. Diese realisieren die in Tabelle 6.8-9 beschriebenen Funktionalitäten. Aus Annotationen können relationale Datenbankschemata generiert werden.

Hibernate-Markierung	Beschreibung
@Entity	Auszeichnung einer Klasse als Entity Bean
@Interitance(strategy = <strategy>)	Spezifikation der Mapping-Strategie bei Vererbung
@Id	• Auszeichnung eines Klassenattributes als eindeutigen Identifikator der Bean • Erforderlich für jede Klasse, die als Entity Bean spezifiziert ist
@OneToOne(cascade = <Cascade-Type>)	Spezifikation einer 1:1-Relation

@ManyToOne()	Spezifikation einer n:1-Relation
@OneToMany(mappedBy = <attribut>, cascade = <CascadeType>)	Spezifikation einer 1:n-Relation
@ManyToMany(targetEntity = <target class>, cascade = <CascadeType>)	Spezifikation einer m:n-Relation

Tabelle 6.8-9: ***Verwendete Hibernate-Annotationen und ihre Bedeutungen***
Quelle: *Bergmann/Klinger/Winkler* (2006, 69)

Das ER-Modell für die EPK-Objektstruktur (processRepository.epk.model) ist in Abbildung A.10-1 dargestellt. Dabei werden die Aggregationen und Vererbungen in der objektorientierten Darstellung der EPK-Objektstruktur im Datenbankschema jeweils über einen Fremdschlüssel abgebildet (Bergmann/Klinger/Winkler 2006, 69). In Abbildung A.10-2 ist das ER-Modell der PML-Repräsentation dargestellt.

6.8.7.3 Prozesssteuerung und Persistenzsicherung zur Laufzeit des Systems

Im Folgenden werden die Details zur Prozesssteuerung beschrieben und dabei die dafür wesentlichen Eigenschaften der Implementierung erläutert, wobei auch auf die Benutzeroberfläche und relevante Teile der Persistenzsicherung eingegangen wird. Zur Prozesssteuerung werden die folgenden Schritte durchgeführt, die in diesem Abschnitt erläutert werden:

- Interpretation des Prozesses
- Abarbeitung des Prozesses einschließlich Persistenzsicherung und Visualisierung an der Benutzeroberfläche

Zur Interpretation des eingelesenen Prozesses (Abbildung A.8-2) im Format PML wird in der Klasse ProcessorFactory für den Wurzelprozess die Methode createFromXMLElement durch den ProcessMasterAgent aufgerufen. In dieser Methode werden dafür ein ProcessorState- und ein ProcessorBehaviour-Objekt, das als Identifikator den im name-Attribut definierten Prozessnamen erhält, für den Wurzelprozess generiert. Zur Interpretation der im process-Element (für die Interpretation der XML-Elemente siehe Tabelle 6.8-10) für den Wurzelprozess referenzierten Elemente process und activity wird in der Klasse ProcessorFactory die Methode parse aufgerufen. Für jedes activity-Element wird die Methode parseActivity aufgerufen, in der ein Objekt für das Behaviour generiert wird, dessen Typ durch die im type-Attribut definierte Klasse bestimmt ist. Weiterhin wird durch die Methode getConstraints das ConstraintSet-Objekt aus dem constraints-Element generiert. In der Methode addActivity werden das soeben erzeugte ConstraintSet-Objekt sowie das Behaviour dem ProcessorBehaviour des Wurzelprozesses übergeben. Die Parameter für die Aktivitäten werden durch die Methode setParameters gesetzt. Dem Behaviour ProcessorBehaviour des Wurzelprozesses stehen damit alle Informationen über auszuführende Behaviours und ihre Vorbedingungen zur Verfügung.

Nach der beschriebenen Interpretation der activity-Elemente werden die process-Elemente in ähnlicher Form interpretiert: Dazu erfolgen rekursive Aufrufe der Methode createFromXML-Element, um die kompositen process-Elemente auf elementare activity-Elemente zu reduzieren.

Für jedes solche Element erfolgen eine Instanzbildung des Behaviours ProcessorBehaviour und die Integration dieses Behaviours in das übergeordnete Behaviour ProcessorBehaviour durch den Aufruf der Methode addActivity.

XML-Element	Interpretation
process	Für jedes process-Element wird ein neues ProcessorBehaviour instantiiert und dem übergeordneten ProcessorBehaviour hinzugefügt.
activity	Die im XML-Attribut type angegebene Klasse (Instanz des IProcessingBehaviours) wird instantiiert und dem übergeordneten ProcessorBehaviour hinzugefügt.
constraints	Werden durch ein ConstraintSet-Objekt repräsentiert
constraint	Wird durch ein Repository-Objekt repräsentiert
agent	Nicht unterstützt
parameter	Wird dem IProcessingBehaviour als XML-Element übergeben

Tabelle 6.8-10: ***Interpretation der unterstützten XML-Elemente***
Quelle: In Anlehnung an *Bergmann/Klinger/Winkler* (2006, 60)

Sobald ein process-Element wie beschrieben interpretiert ist, erfolgt ein Vergleich mit denjenigen Einträgen in der Datenbank, die den Zustand dieses Prozessschrittes speichern. Über die update-Methode in der Klasse ProcessState kann der ggf. bereits in der Datenbank gespeicherte Zustand des Prozesses wiederhergestellt werden, um ihn an der gespeicherten Stelle fortzusetzen.

Die Prozessabarbeitung zur Laufzeit wird durch die Benutzeroberfläche (Start-Button, siehe Abbildung 6.8-22 und die Ausführungen zur Benutzeroberfläche weiter unten) gestartet und durch den Agenten ProcessMasterAgent mit seinem Behaviour ProcessorBehaviour realisiert. Die in letzterem verwalteten Behaviours sind die aus Unterprozessen (process-Elemente) abgeleiteten ProcessorBehaviours oder die durch die activity-Elemente spezifizierten Behaviours, die tatsächliche Aktivitäten der Agenten beschreiben. ProcessorBehaviour ist eine Spezialisierung der JADE-Klasse CompositeBehaviour, die das Scheduling des nächsten auszuführenden Behaviours umsetzt. Folglich müssen in der Klasse ProcessorBehaviour die folgenden im CompositeBehaviour abstrakt deklarierten Methoden implementiert werden, deren Funktionalität im Weiteren dargestellt wird:

- scheduleFirst
- scheduleNext
- checkTermination
- getCurrent

- getChildren[91]

Zur Abarbeitung des eingelesenen Prozesses werden die im Folgenden beschriebenen Schritte ausgeführt (siehe auch Abbildung A.8-1): In der Methode scheduleNext wird überprüft, ob das aktuell auszuführende Behaviour bereits vollständig verarbeitet ist. Ist dies der Fall, wird die Methode onCurrentDone der Klasse ProcessorBehaviour aufgerufen. In dieser Methode wird das ProcessorState-Objekt zur Speicherung des Zustandes der gesamten Prozessabarbeitung aktualisiert. Dazu wird das Ergebnis des aktuellen Behaviours in der Datenbank gespeichert. Weiterhin wird das aktuelle Behaviour aus dem Objekt ProcessorState und ProcessingState entfernt.

Um wartende Behaviours zu aktivieren, wird zunächst in der Methode checkBlockedBehaviours überprüft, ob Vorbedingungen zur Ausführung eines der Behaviours erfüllt sind. Dazu werden die Constraints der Behaviours mit dem Objekt fullFilledConstraints vom Typ Repository in der Klasse ProcessorState verglichen. Kann eines der überprüften Behaviours ausgeführt werden, wird das entsprechende ProcessState-Objekt aktualisiert, das gewählte Behaviour wird aus der Liste der wartenden Behaviours entfernt und mit dem Aufruf der Methode setCurrent in der Klasse ProcessorState als nächstes ausführbares Behaviour gekennzeichnet. Zur Aktualisierung der Benutzeroberfläche (siehe dazu die Ausführungen weiter unten) werden die Methoden activityFinished und onSchedule der Schnittstelle IProcessorController aufgerufen.

Über den Aufruf der Methode getCurrent in der Klasse ProcessorState wird das eben als ausführbar gekennzeichnete Behaviour mit dem Aufruf der Methode action aktiviert. Mit diesem Aufruf erfolgt die Abarbeitung der eigentlichen Funktionalität, die durch das entsprechende Behaviour des Agenten bestimmt ist.

Nach der Abarbeitung eines Behaviours wird die Methode checkTermination in der Klasse ProcessorBehaviour aufgerufen. Sind alle Behaviours abgearbeitet, wird zur persistenten Speicherung des aktuellen Status des Prozesses die Methode saveState der Schnittstelle IProcessPersister aufgerufen (siehe dazu die Ausführungen zur Persistenz weiter unten). Wenn alle im process-Element als Wurzel des Gesamtprozesses verwalteten Behaviours abgearbeitet sind, ist die Abarbeitung des gesamten Prozesses beendet.

Werden zur Abarbeitung des Prozesses die Methoden scheduleFirst oder scheduleNext in der Klasse ProcessorBehaviour aufgerufen, wird der Controller IProcessorController durch die Aktivierung seiner Methoden activityFinished und onSchedule über die Klasse ProcessorBehaviour benachrichtigt (Bergmann/Klinger/Winkler 2006, 67). Wird ein Prozessschritt erfolgreich ausgeführt, wird über activityFinished die Methode currentExecuted in der Klasse ProcessorTreeModel aufgerufen. Darin wird das aktuell beendete Behaviour aus der als Model-Komponente realisierten Datenstruktur ProcessorTreeModel entfernt. Über die View-Komponente ProcessorIntFrame, welche als Listener am Modell ProcessorTreeModel registriert ist, wird dieser Schritt entsprechend dem MVC-Prinzip an die Benutzerschnittstelle propagiert. Dabei wird aus der

[91] Über die Methode getChildren werden alle in der Klasse ProcessorBehaviour verwalteten Behaviours des Agenten ProcessMasterAgent zurückgeliefert. Der Aufruf dieser Methode erfolgt bei der Aktualisierung der Benutzerschnittstelle zur Visualisierung des Behandlungsprozesses in einer textuellen und baumartigen Struktur (siehe dazu die Ausführungen weiter unten in diesem Abschnitt).

Baumdarstellung des Prozesses (siehe Abbildung 6.8-22) der gerade abgearbeitete Prozessschritt gelöscht.

Über die Methode onSchedule wird die schrittweise Abarbeitung des gesamten Prozesses umgesetzt (Bergmann/Klinger/Winkler 2006, 67): Wenn diese Methode aufgerufen wird, wird der aktuell ausgeführte Thread über seine Methode wait in den Wartezustand versetzt. Über die Methode notify wird der Thread an dieser Stelle wieder fortgeführt, um nur den nächsten Prozessschritt auszuführen.

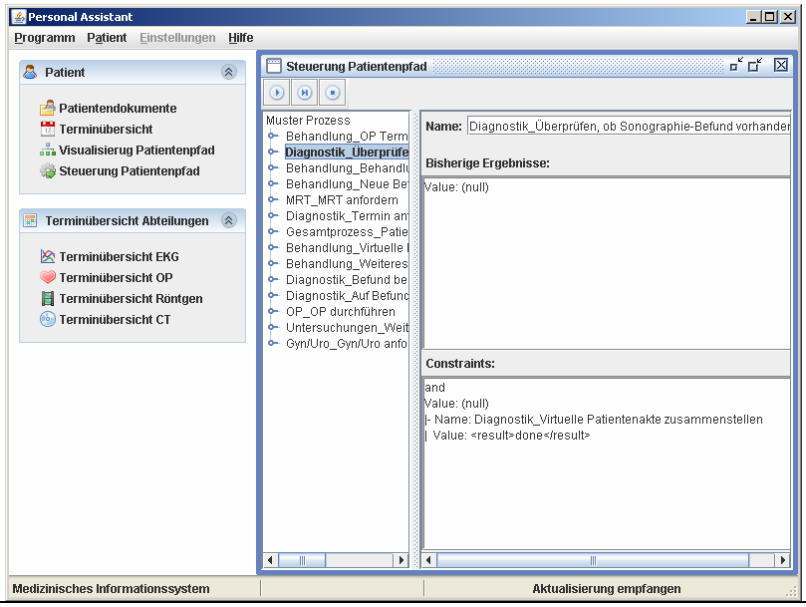

Abbildung 6.8-22: ***Textuelle Darstellung eines exemplarischen Behandlungsprozesses***[92]
Quelle: Eigene Darstellung

Um die Persistenz bei der Prozessverarbeitung zu gewährleisten, werden für jeden Prozessschritt (ProcessorBehaviour) in der darin referenzierten Klasse ProcessorState die folgenden Eigenschaften gespeichert (Bergmann/Klinger/Winkler 2006, 64-65):

- Familie der jeweils verwalteten Behaviours
- Vorbedingungen dieser Behaviours

[92] Der dargestellte Prozess wurde im Rahmen der Arbeiten bei *Bergmann/Klinger/Winkler* (2006) für ausgewählte Anforderungen der vorliegenden Arbeit entworfen. Zur Reduktion der Implementierungskomplexität wurde der Behandlungsprozess fiktiv und nicht entsprechend den Abläufen des in Abschnitt 6 beschriebenen Szenarios modelliert. Der exemplarische Prozess wurde explizit so konstruiert, dass die Interaktionen der agentenbasierten Prozesssteuerung mit der Agentenanwendung zur Dokumentenaggregation (siehe Abschnitte 6.7.1 und 6.8.1) sowie mit dem Simulationssystem SeSAm (siehe dazu Abschnitt 6.8.8) umgesetzt werden. Somit wird gezeigt, dass das AMD-Konzept eine geeignete Schnittstelle besitzt, die die Aggregation der Patientendokumente durch die Prozesssteuerung ermöglicht. Die Eigenschaften von SeSAm (siehe dazu Abschnitt 6.8.8) erlauben es in einem weiteren Schritt, das durch SeSAm simulierte Terminplanungssystem mit der in den Abschnitten 6.7.4 und 6.8.5 beschriebenen Scheduling-Applikation zu ersetzen. Insofern wird die Qualität der Forschungsergebnisse durch diese Einschränkung gegenüber dem in Abschnitt 6.6.1 beschriebenen Prozessausschnitt nicht reduziert, weil dort ebenso die Dokumentenaggregation sowie die Anbindung an ein Terminplanungssystem involviert sind.

- Menge der wartenden Behaviours
- Gegenwärtig auszuführendes Behaviour

Die in ProcessorState referenzierten Behaviour-Objekte können nicht unmittelbar über eine Hibernate-Abbildung in die Persistenzschicht übertragen werden, weil dabei nur Objekte mit primitiven Typen abgelegt werden können (Bergmann/Klinger/Winkler 2006, 65). Deshalb werden für eine Persistenz-Abbildung in der Klasse ProcessState die folgenden primitiven Werte gespeichert (Bergmann/Klinger/Winkler 2006, 65):

- Identifikatoren der jeweils verwalteten Behaviours
- Menge der Identifikatoren für die wartenden Behaviours
- Menge der erfüllten Vorbedingungen

Die komposite Struktur von ProcessorBehaviour-Objekten wird für die Persistenzabbildung so umgesetzt, dass jedes ProcessState-Objekt dasjenige des übergeordneten ProcessorBehaviour-Objekts referenziert (Bergmann/Klinger/Winkler 2006, 65). Im Fall des Wurzelprozesses ist eine solche Referenz auf das übergeordnete ProcessorBehaviour-Objekt nicht gegeben. Mit diesen Implementierungsentscheidungen wird gewährleistet, dass die in der Datenbank abzulegenden Objekte vom Typ ProcessState lediglich primitive Typen besitzen oder auf andere Objekte verweisen, die ebenso nur primitive Typen kapseln.

Die Persistenzsicherung wurde entsprechend dem Transaktionskonzept (siehe dazu z.B. Weikum/Vossen 2002) umgesetzt (Bergmann/Klinger/Winkler 2006, 66): Die Granularität der Transaktionen umfasst vollständige Prozessschritte. Nach der erfolgreichen Ausführung eines solchen Schrittes werden die Ergebnisse in der Persistenzschicht festgeschrieben (Methode saveState in Abbildung A.8-1). Eine Wiederaufnahme ist dementsprechend nach der letzten erfolgreichen Ausführung eines Prozessschrittes möglich. Mit diesem Mechanismus ist neben der Abdeckung von Fehlerfällen auch die Zwischenspeicherung von Behandlungspfaden möglich. Bei der Speicherung wird das ProcessState-Objekt des aktuell und erfolgreich ausgeführten Prozessschrittes in der Persistenzschicht abgelegt. Die weiter oben beschriebene komposite Datenstruktur sowie die Hibernate-Kaskadierungspotenziale erlauben mit dieser Speicherung die Persistenzsicherung eines vollständigen Prozesses (Bergmann/Klinger/Winkler 2006, 66). Damit werden die Implementierungsvorteile demonstriert und letztlich auch die Wahl für Hibernate begründet.

Zur persistenten Speicherung von Informationen zur Prozessverarbeitung werden, wie weiter oben beschrieben, Objekte vom Typ ProcessState über Hibernate in einer relationalen Datenbank abgelegt (Bergmann/Klinger/Winkler 2006, 67). Die Abbildung dieser Klasse in Relationen wird in der vorliegenden Arbeit, wie in Abschnitt 6.8.7.2 beschrieben, über Annotationen im Programmtext realisiert. Das aus diesen Annotationen generierte relationale Datenbankschema ist in Abbildung A.10-3 dargestellt.

Zum Zeitpunkt der Interpretation einer PML-Beschreibung werden die Zustände der ProcessorState-Objekte mit den gespeicherten ProcessState-Zuständen aus der Persistenzschicht ergänzt (Methode updateState in Abbildung A.8-2), falls solche bereits gespeichert sind (Bergmann/Klinger/Winkler 2006, 65). Sind diese Objekte noch nicht gespeichert, werden die

generierten ProcessState-Objekte in der Datenbank abgelegt, um den aktuellen Zustand des Prozesses nach dem Einlesen der Prozessinformationen zu speichern.

Zur Rückkopplung der Ergebnisse der Prozesssteuerung an die Benutzeroberfläche werden die folgenden Funktionalitäten realisiert: Nach der Auswahl des gewünschten Behandlungsprozesses wird von der Ereignisbehandlung die Methode executeProcess aufgerufen. Darin werden u.a. dem neu generierten Objekt CompositeProcessorController die beiden Controller UIProcessorController und GraphUIProcessorController übergeben, welche jeweils die Schnittstelle IProcessorController implementieren. Ersterer ist an die Darstellung von Prozessen an der Benutzeroberfläche zur Prozesssteuerung gebunden (siehe Abbildung 6.8-22), letzterer visualisiert den aktuellen Prozessfortschritt durch eine EPK-Darstellung (Abbildung 6.8-23). Zur Realisierung der jeweiligen Funktionalitäten werden die Methodenaufrufe von activityFinished und onSchedule vom CompositeProcessorController an die jeweils registrierten Controller zur Weiterverarbeitung delegiert.

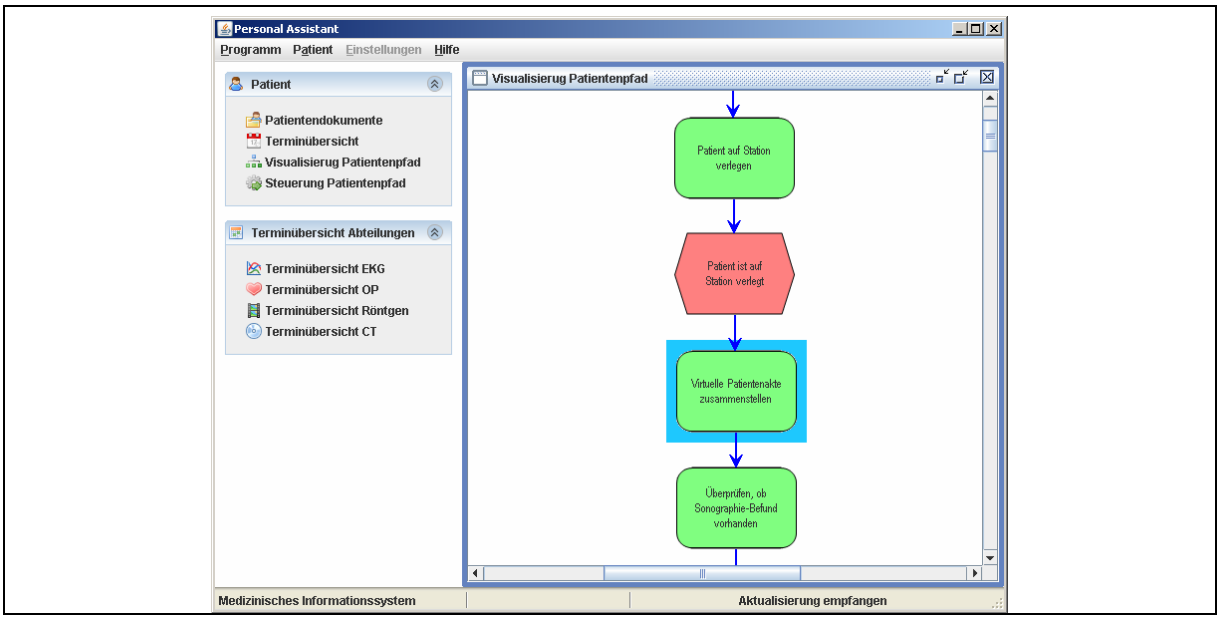

Abbildung 6.8-23: ***Benutzeroberfläche mit der EPK-Darstellung des gewählten Behandlungsprozesses***[93]
Quelle: Eigene Darstellung

Im Konstruktor von GraphUIProcessorController wird u.a. eine Instanz der Klasse VisualizationGraph gebildet, über die der aktuell verarbeitete Prozess aus der Datenbank ausgelesen und mit der Bibliothek JGraph[94] als EPK visualisiert wird (Slavtchev 2007, 28-32). Dabei wird insbesondere der aktuell bearbeitete Prozessschritt hervorgehoben (siehe Abbildung 6.8-23).

[93] Siehe dazu die Anmerkungen in Fußnote 92.

[94] Weitere Informationen über diese Bibliothek zur Visualisierung von Graphen können auf den folgenden Seiten abgerufen werden: http://www.jgraph.com, zugegriffen am 10.07.2007. In dem der vorliegenden Arbeit zugrunde liegenden Forschungsprojekt wurde vom Hersteller die Erweiterung JGraph Layout Pro zur Verfügung gestellt, die Layoutalgorithmen für die Visualisierung von Graphen realisiert. In der Arbeit von *Slavtchev* (2007, 14-17) wurde allerdings festgestellt, dass die verfügbaren Algorithmen eine Vi-

Die Ausführungen zur Implementierung der Prozessunterstützung demonstrieren, dass diese Funktionalität geeignet durch ein agentenbasiertes System realisiert werden kann. Insbesondere werden dabei auch Erkenntnisse und Standards zur Workflow-Unterstützung berücksichtigt und an agentenbasierte Systeme angepasst (Krcmar/Horn 2003, 23-24). Insgesamt kann mit der Beschreibung in Abschnitt 6.8.7 die Anforderung 9 aus Abschnitt 6.4.1 erfüllt werden.

Weiterhin kann festgestellt werden, dass mit der dargestellten Implementierung Teilprozesse zu einem Gesamtprozess zusammengeführt werden können. Damit wird das Prinzip der Prozessintegration aus der EAI umgesetzt. Außerdem wird neben der Datenaggregation mit der Prozessunterstützung weitere Funktionalität in dem AMD zur Verfügung gestellt. Diese Funktionalität orientiert sich an der Fokussierung von relevanten Informationen für einen Behandlungsprozess und erfüllt somit das wesentliche Kriterium eines Dienstes im Sinne einer SOA. Damit wird Anforderung 15 aus 6.4.2 Abschnitt erfüllt, welche die Berücksichtigung von EAI- und SOA-Prinzipien verlangt.

6.8.8 Integration mit weiteren Anwendungen

Die bisherigen Ausführungen zur Beschreibung des Designs und der Implementierung der agentenbasierten Applikation zeigen, dass auf der Grundlage des AMD unterschiedliche Anwendungen integriert werden können. Um den Entwicklungsprozess zur Integration von nicht agentenbasierten Anwendungen in ein Agentensystem zu unterstützen, können externe Systeme zunächst über ein Simulationssystem (SeSAm, Shell for Simulated Agent Systems, siehe dazu z.B. Heine/Herrler/Kirn 2005) integriert werden. Die sukzessive Integration von Informationssystemen in das Agentensystem wird damit vereinfacht. Die folgenden Ausführungen fassen die Beschreibung bei *Bergmann/Klinger/Winkler* (2006, 73-75 und 83-85) anhand eines exemplarischen Prozesses[95] zusammen:

Aus architektonischer Perspektive ist zu erwähnen, dass die Kommunikation zwischen einer agentenbasierten Anwendung und dem genannten Simulationssystem über die Registrierung eines Gateway-Agenten von SeSAm im DF der Agentenanwendung ermöglicht wird. Für diesen Agenten wird eine weitere Agentenplattform aus der SeSAm-Anwendung gestartet. Dieser Agent übernimmt neben der Kommunikation mit dem agentenbasierten System auch diejenige mit der Verarbeitungsvorschrift in der Simulationsumgebung.

In dem genannten Beispielprozess wird von SeSAm ein Terminvergabesystem simuliert. Die darin definierte Verarbeitungsvorschrift führt die folgenden Schritte aus (Bergmann/Klinger/Winkler 2006, 74):

- Warten auf spezifizierte Nachrichten (zyklisches Behaviour)
- Empfang einer Nachricht
- Verarbeitung der Nachricht
- Auswahl eines geeigneten Termins[96]
- Konstruktion einer Antwortnachricht

---

sualisierung von EPKen nicht geeignet unterstützen. Für die Erstellung eines adäquaten Layouts sind deshalb weitere Arbeiten erforderlich.

[95] Siehe dazu die Anmerkungen in Fußnote 92.

[96] In der vorliegenden Arbeit wird ein fest vorgegebener Termin zurückgeliefert. Diese Implementierung ist für die gegebenen Simulationszwecke ausreichend. Für eine Weiterentwicklung der prototypischen Implementierung ist an dieser Stelle das in Abschnitt 6.8.5 beschriebene Scheduling-System zu integrieren.

- Versand der Antwort an den Anfrager

Der in SeSAm simulierte Agent SESAM_APPOINTMENT_AGENT setzt diese Verarbeitungsvorschrift um und wird über einen entsprechenden Gateway-Agenten im DF der Umgebung Hospital repräsentiert, um die von ihm angebotenen Dienste zur Verfügung zu stellen.

Die Integration der Simulationsumgebung in die in der vorliegenden Arbeit konstruierte Agentenanwendung wird im Folgenden beschrieben: Terminanfragen werden von der Prozesssteuerung über den AppointmentManager an SeSAm versendet. Dazu wird an der entsprechenden Stelle in dem Beispielprozess vom Behaviour RequTerminBeh des ProcessMasterAgent eine Nachricht mit der folgenden Spezifikation an den AppointmentManager gesendet:

- Performativ REQUEST
- ConversationID Sesam-Appointment-Request

Das Behaviour ListenForSesamAppointmentRequst des AppointmentManagers, welches die Schnittstelle IListenForAppointmentBehaviour implementiert, nimmt diese Nachricht entgegen und leitet sie über das Behaviour SendAppointmentRequestToSeSAm an das durch SeSAm simulierte Terminverhandlungssystem weiter. Dort wird die weiter oben beschriebene Verarbeitungsvorschrift ausgeführt. Nach ihrer Abarbeitung empfängt der AppointmentManager die Nachricht von SeSAm über das Behaviour ReceiveAppointmentAnswerFromSeSAm und leitet sie über das Behaviour SendAppointmentAnswerFromSeSAm an den ursprünglich anfragenden Agenten, d.h. den ProcessMasterAgent, weiter. Im Behaviour RecTerminBeh wird im ProcessMasterAgent die vom AppointmentManager weitergeleitete Nachricht empfangen und der ggf. übergebene Termin entnommen und als Ergebnis des Teilschrittes festgehalten (Bergmann/Klinger/Winkler 2006, 85):

- „Appointment Accepted“ <Termin> oder
- „None of the proposed appointments were accepted“

Die Ausführungen in diesem Abschnitt demonstrieren, dass mit SeSAm eine sukzessive Einbettung nicht agentenbasierter Anwendungen in ein MAS unterstützt wird. Insbesondere konnte dabei gezeigt werden, dass nicht notwendigerweise agentenbasierte Applikationen in die implementierte Prozessunterstützung integriert werden können.

Weiterhin wird mit der Beschreibung dargestellt, wie die grundlegende Integration der Scheduling-Anwendung aus Abschnitt 6.8.5 in einen Behandlungsprozess bzw. die entsprechende Prozesssteuerung erfolgen kann. Somit kann insgesamt Anforderung 17 aus Abschnitt 6.4.2 erfüllt werden.

Außerdem konnte die Interoperabilität der in der vorliegenden Arbeit beschriebenen Implementierung mit einer anderen Agentenanwendung demonstriert werden, womit die Offenheit des Prototyps impliziert wird.

6.8.9 Zusammenfassung der Implementierungsbeschreibung

Die Implementierung, welche aus der Designbeschreibung in Abschnitt 6.7 abgeleitet ist, wurde in den Abschnitten 6.8.1 mit 6.8.7 zusammenfassend beschrieben. Dabei konnte demonstriert werden, dass Agentensysteme mit der AMD-Architektur dazu geeignet sind, wesentliche Anforderungen aus dem Gesundheitswesen zu erfüllen. Insbesondere wurden alle in Abschnitt 6.3.6 beschriebenen Anforderungen des konkretisierten Kriterienkatalogs erfüllt.

Damit kann die in Abschnitt 1.4 angegebene Forschungsfrage 3 als beantwortet betrachtet werden.

6.9 Zusammenfassung

Ausgehend von der Argumentation, dass Software-Agenten für Problemstellungen im Gesundheitswesen vorteilhaft eingesetzt werden können (siehe dazu Abschnitt 5), wird in Abschnitt 6 ein agentenbasiertes System prototypisch konstruiert. Grundlage für die Implementierung ist ein ausgewähltes Szenario in einem Referenzkrankenhaus (Abschnitt 6.1), in das das Software-System eingebettet ist. Für die Implementierung wird eine Reihe von technischen Rahmenbedingungen berücksichtigt, die in Abschnitt 6.3 beschrieben werden. Dabei wird insbesondere auf die Vorteilhaftigkeit der Java-Plattform sowie des eingesetzten JADE-Rahmenwerks eingegangen. Letzteres bietet durch Erweiterungen die Möglichkeit, die besonderen Anforderungen der Implementierung zu erfüllen, wie z.B. die Bereitstellung von Agentendiensten über Web Services oder die Portierung von agentenbasierten Anwendungen auf mobile Endgeräte.

Der abstrakte Anforderungskatalog aus Abschnitt 5.4 wird in Abschnitt 6.3.6 konkretisiert und in funktionale sowie nicht funktionale Anforderungen klassifiziert. Diese Anforderungen werden durch das in Abschnitt 6.7 beschriebene Design der Anwendung bzw. die in Abschnitt 6.8 dargestellte Implementierung erfüllt. Somit können die Forschungsfragen 2 und 3 aus Abschnitt 1.4 beantwortet werden: Mit dem AMD wird ein geeignetes Lösungskonzept beschrieben, das die Grundlage für die Implementierung einer modularen Komponente bildet. Das AMD ist insbesondere in der Lage, unterschiedliche Anwendungen unter der Berücksichtigung der Fokussierung des Patienten, seiner Daten und des assoziierten Behandlungsprozesses zu vereinen. Mit der dadurch induzierten Daten- und Prozessintegration sowie der Bereitstellung von Diensten im Sinne der SOA können wesentliche Eigenschaften der EAI bzw. SOA umgesetzt werden.

7 Bewertung der Ergebnisse

7.1 Überblick

In Abschnitt 7 (Abbildung 7.1-1, für die Einordnung in den Gesamtzusammenhang siehe Abbildung 1.12-1) werden die in den bisherigen Ausführungen beschriebenen Darstellungen bewertet. Dazu wird zunächst ein Evaluierungsrahmen erläutert, anhand dessen die Evaluierung erfolgt. Ergänzt werden diese Aussagen durch eine Beschreibung der Ergebnisse von Labortests. Außerdem werden die in Abschnitt 6 beschriebenen Applikationen einzeln bewertet. Auf der Grundlage dieser Ausführungen werden die Ergebnisse der vorliegenden Arbeit hinsichtlich Entwicklungsperspektiven für KISe sowie eines Mehrwertes durch AMDe dargestellt. Damit ist insbesondere Forschungsfrage 4 aus Abschnitt 1.4 zu beantworten.

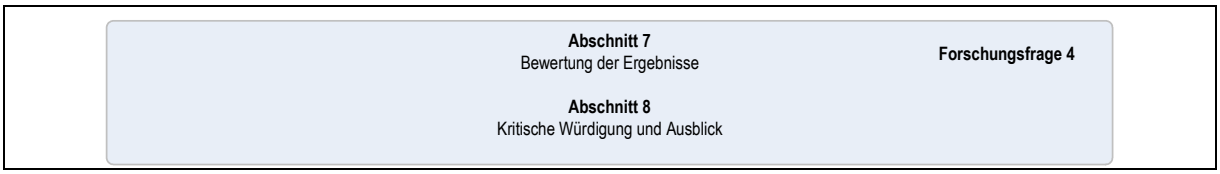

Abbildung 7.1-1: ***Überblick über Abschnitt 7***
Quelle: Eigene Darstellung

7.2 Evaluierungsrahmen

Ziele einer Evaluierung sind die Überprüfung des Einsatzpotenzials für einen Ansatz unter bestimmten Bedingungen und die Entscheidungsunterstützung für die Auswahl einer geeigneten Lösung für ein gegebenes Problem (Zöller et al. 2006, 558). In der vorliegenden Arbeit soll die Eignung der gewählten Lösung für die identifizierten Problemstellungen im Gesundheitswesen gezeigt werden. Für die Evaluierung werden dazu die folgenden Aspekte betrachtet (Zöller et al. 2006, 558-559):

- *Bezugspunkt*: Im Gegensatz zur komparativen Evaluierung[97] (Benchmarking) wird bei der deskriptiven Evaluierung kein vergleichendes System herangezogen. Die Bezugspunkte werden in den folgenden Ausführungen für die vorliegende Arbeit bestimmt.
- *Zeitpunkt*: In Abhängigkeit vom Zeitpunkt kann zwischen formativer und summativer Evaluierung differenziert werden (Scriven 1980). Formative Evaluierung erfolgt in einem zyklischen Prozess und kann damit als Eingabe für eine erneute Iteration der Entwicklung fungieren. Die Evaluierung am Ende eines Prozesses wird als summativ bezeichnet. Für die vorliegende Arbeit kann die Bewertung nur summativ erfolgen, weil der entwickelte Prototyp nicht unter realen Einsatzbedingungen eingesetzt wird und somit keine Erkenntnisse aus der Domäne gewonnen werden können, die in die weitere Entwicklung einfließen können.
- *Methode*: Unterschieden werden kann zwischen der Evaluierung in einem Labor bzw. der realen Welt (Dix 1998). Da der implementierte Prototyp nicht den Anspruch an die Erfüllung von realen Einsatzbedingungen erhebt, wird in der vorliegenden Arbeit die

[97] *Camp* (1989; 1995), *Rechenberg/Pomberger* (1997), *Heinrich/Heinzl/Roithmayr* (2004).

Evaluierung im Labor durchgeführt (siehe dazu die Ausführungen zu Labortests in Abschnitt 7.4).

- *Fokus*: In Abhängigkeit von der Betrachtung des gesamten Systems oder nur von Teilen wird die Evaluierung auf Makro- bzw. Mikroebene durchgeführt. Die Evaluierung erfolgt im Folgenden durch die Betrachtung auf den Ebenen Mikro-, Meso- und Makroebene.

Die genannten Aspekte können als Konkretisierungen eines Evaluierungsmodells (siehe auch Abbildung 7.2-1) betrachtet werden (Zöller et al. 2006, 558). Die Evaluierung nach diesem Modell wird im Folgenden beschrieben (Zöller et al. 2006, 560):

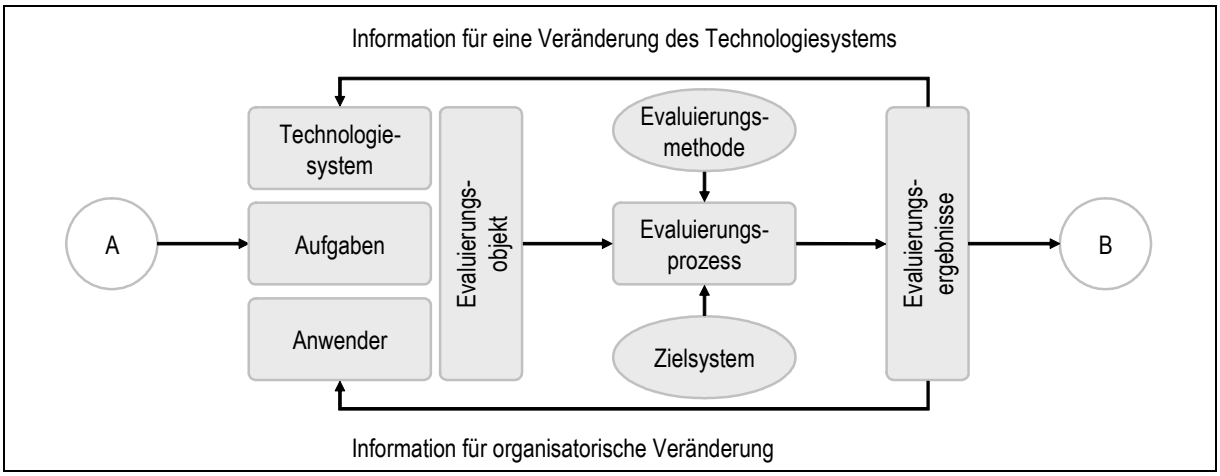

Abbildung 7.2-1: ***Evaluierungsmodell***
Quelle: In Anlehnung an *Gappmair/Häntschel* (1997, 66)

Als Evaluierungsobjekte dienen üblicherweise das Technologiesystem, die Anwender und die von ihnen durchzuführenden Aufgaben. In der vorliegenden Arbeit wird das entwickelte System fokussiert, weil letzteres nicht in einer produktiven Umgebung eingesetzt wird. Im Evaluierungsprozess wird das gewählte Objekt durch die Evaluierungsmethode mit den Kriterien des Zielsystems verglichen. Die daraus gewonnenen Evaluierungsergebnisse können als Information für die potenzielle Veränderung oder Weiterentwicklung der Evaluierungsobjekte eingesetzt werden. Im Zusammenhang mit der vorliegenden Arbeit werden diese Erkenntnisse für potenzielle Veränderungen als Ausblick (siehe Abschnitt 8.3) dargestellt.

Mit diesem Evaluierungsmodell als Grundlage werden für die weiteren Ausführungen die in Tabelle 7.2-1 dargestellten Aspekte (Zöller et al. 2006, 567) zur Evaluierung der technologischen Eigenschaften der prototypischen Implementierungen und der Charakteristika der Domäne herangezogen (Zöller et al. 2006, 567-569, ergänzt durch eigene Ableitungen). Die Ergebnisse der Evaluierung entlang dieser Aspekte werden in den folgenden Abschnitten beschrieben.

Evaluierungsaspekt	Technologische Eigenschaft der Implementierung	Domäneneigenschaft
Evaluierungsobjekte	Systemeigenschaften der Anwendung zur Erfüllung der identifizierten Anforderungen	Abbildung von Domäneneigenschaften in einem agentenbasierten System
Evaluierungskriterien	Korrektheit der Implementierung bzw. Algorithmen	Erfüllung der Abbildung von Eigenschaften
Evaluierungsmethode	Testen, Vergleich	Analytische Untersuchung
Evaluierungsreferenz	Anforderungskatalog	Status quo der Domäne

Tabelle 7.2-1: ***Überblick über Evaluierungsaspekte und ihre Technologie bzw. Domänen bezogenen Ausprägungen***
Quelle: *Zöller* (2006, 567-569), ergänzt um eigene Erweiterungen

7.3 Evaluierung

7.3.1 Evaluierung der technologischen Eigenschaften

Für die Evaluierung der technologischen Eigenschaften werden die implementierten Systemeigenschaften auf ihre Korrektheit untersucht. Dabei werden die Charakteristika den Aspekten aus dem Anforderungskatalog (siehe Abschnitte 4.12, 5.4 sowie 6.3.6) gegenüber gestellt.

Bereich aus dem abstrakten Anforderungskatalog aus Abschnitt 4.12 (Evaluierungsreferenz)	Realisierung der Anforderungen aus Abschnitt 5.4 (Evaluierungsreferenz)	Konkretisierung der Anforderungen aus Abschnitt 6.3.6 (Evaluierungsobjekt)
Plattform (Anforderung 1)	• Einbettung in die TI • Berücksichtigung heterogener Plattformen und Endgeräte	• Simulation der Elemente Konnektor (Anforderung 1) und Kartenlesegerät (Anforderung 2) • Speicherung der letzten 50 Zugriffe auf die eGK (Anforderung 3) • Realisierung der Anwendung auf der Basis der Java-Laufzeitumgebung (Anforderung 12) und der FIPA kompatiblen Agentenplattform JADE (Anforderung 19) • Portierung des Prototyps für eine virtuelle Patientenakte auf ein mobiles Endgerät (Anforderungen 13 und 19) unter Berücksichtigung der beschränkten Ressourcen des Endgerätes durch die Konfiguration der Java-Laufzeitumgebung und Gestaltung einer geeigneten Benutzeroberfläche (Anforderung 14)
Syntax (Anforderung 2)	Unterstützung des XML basierten Standards HL7 CDA	Versand der aggregierten Daten im Format HL7 CDA (Anforderung 4)

Kontext (Anforderung 3)	Eindeutige Patientenidentifikation	• Ausarbeitung und Umsetzung des Konzeptes der aktiven medizinischen Dokumente • Kapselung der zu einem Patienten gehörigen Informationen in einem kompositen Software-Agenten (Anforderung 5) entsprechend der EAI-Datenintegration (Anforderung 15)
Semantik (Anforderung 4)	Einbettung von LOINC-Codes in die ausgetauschten Dokumente	• Einbettung von LOINC-Codes in die HL7 CDA-Dokumente (Anforderung 6) • Geeignete Abbildung von Nachrichteninhalten auf Ontologiekonzepte (Anforderung 7)
Anwendung (Anforderung 5)	• Informationsintegration aus verteilten Informationssystemen • Prozesssteuerung • Entscheidungsunterstützung • Integration eines ausgewählten KIS	• Aggregation von Daten aus verteilten Informationssystemen • Ausgeprägte lose Kopplung in einem Agentensystem begünstigt die Integration • Extraktion von Daten aus den Informationssystemen SAP IS-H bzw. i.s.h.med (Anforderung 8) • Extraktion von Daten aus einem (simulierten) PVS • Implementierung einer agentenbasierten Prozessunterstützung als aktive Funktionalität des AMD (Anforderung 9) • Prozessintegration im Sinne der EAI (Anforderung 15) • Entscheidungsunterstützung entlang eines ausgewählten klinischen Pfades (Anforderung 9) • Modularisierte Umsetzung des AMD zur Integration unterschiedlicher Funktionalitäten (Anforderung 17) • Berücksichtigung der Architekturprinzipien SOA/EAI (Anforderung 15), Schichtenarchitektur (Anforderung 16), Modularisierung zur Integration eines Simulationssystems und Patientenportals (17), flexible Komposition und Dekomposition (18)
Umwelt (Anforderung 6)	• Flexible und automatisierte Terminplanung • Berücksichtigung der Patientenorientierung	• Implementierung eines Scheduling-Algorithmus für Termine (Anforderung 10) • Service orientierte Ankopplung eines Patientenportals im Sinne eines eKiosks (11) an das AMD über Web Services (Anforderungen 17 und 19) • Verschlüsselung und Signatur der Kommunikationsflüsse zwischen Konnektor und Kartenlesegerät (Anforderungen 19 und 20)

Tabelle 7.3-1: ***Bewertung von technologischen Eigenschaften***
Quelle: Eigene Darstellung

In Tabelle 7.3-1 sind die Ergebnisse der Evaluierung aus technologischer Perspektive vergleichend zusammengefasst. In Spalte 1 sind dabei die Aspekte aus dem Anforderungskatalog aus Abschnitt 4.12 beschrieben. Die Realisierung der Anforderungen (Abschnitt 5.4) für die prototypische Implementierung als agentenbasiertes System ist in Spalte 2 dargestellt. Mit diesen Kriterien der Anforderungskataloge wird die Evaluierungsreferenz gebildet. In Spalte 3 wird angegeben, wie die Anforderungen (Abschnitt 6.3.6) in dem Anwendungsfall konkretisiert werden, womit die Systemeigenschaften als Evaluierungsobjekte gegeben sind.

Zusammenfassend kann für die beschriebenen Anforderungen festgestellt werden, dass mit der prototypischen Implementierung die Tragfähigkeit der erarbeiteten Konzepte demonstriert werden kann. Aus diesem Vergleich zwischen den Anforderungen und den Umsetzungen kann als Evaluierungsmethode somit die Korrektheit der implementierten Funktionalitäten als Evaluierungskriterium abgeleitet werden. Für die weitergehende Evaluierung mit der Methode Testen wird auf Abschnitt 7.4 verwiesen.

7.3.2 Evaluierung der Domäneneigenschaften

Für die Evaluierung von Domäneneigenschaften ist zu zeigen, dass die Anforderungen und Charakteristika der Domäne (Evaluierungsreferenz) geeignet durch Konstrukte der Software-Technik abgebildet werden können. Damit besitzt die Realisierung durch ein agentenbasiertes System die Rolle des Evaluierungsobjektes. Grundlage für diese Darstellung sind die Ausführungen in Abschnitt 5.1. Die Ergebnisse der Gegenüberstellung der Domäneneigenschaften und der Abbildung auf Software-Ebene sind in Tabelle 7.3-2 zusammengefasst. In dieser Darstellung wird deutlich, dass sich die Problemstellungen des Gesundheitswesens vorteilhaft durch agentenbasierte Systeme lösen lassen.

Domäneneigenschaften (Evaluierungsreferenz)	**Realisierung im agentenbasierten System (Evaluierungsobjekt)**
Vielzahl unterschiedlicher Akteure	• Abbildung auf einzelne Software-Agenten • Skalierbarkeit ist aus technischer Hinsicht gegeben
Komplexe Interaktionen zur Kommunikation, Kooperation und Koordination	Abbildung von Interaktionen zwischen Agenten mit unterschiedlicher Bedeutung: • Methodenaufrufe • Semantisch reichhaltige Interaktionen • Protokoll basierte Kommunikation
Emergenz durch a priori nicht vollständig determinierte Prozesse und Planungen	Flexible Reaktionsfähigkeit von Agenten
Ambiente Intelligenz	• Persönliche und Sichten abhängige Assistenz für unterschiedliche Akteure • Repräsentierung durch die (potenziell gleichzeitige) Annahme von geeigneten Rollen • Prozessorientierung des AMD
Verteiltheit im Gesundheitswesen	Einbettung in eine geeignete Plattform (TI)
Informationslogistik	• Aggregation von Daten aus verteilten Informationssystemen • Anonymisierte Bereitstellung von Daten • Besonders ausgeprägte lose Kopplung in einem agentenbasierten System unterstützt die Integration • Flexibilität zur Komposition und Dekomposition von Subsystemen

Tabelle 7.3-2: ***Bewertung von Domäneneigenschaften***
Quelle: Eigene Darstellung

7.3.3 Evaluierung aus Anwender- und Aufgabenperspektive

Mit der angestrebten Ausrichtung von Prozessen am Patienten ergeben sich für die Anwendergruppen Patienten und Leistungserbringer die im Folgenden beschriebenen potenziellen Veränderungen:

Patienten erhalten die bei *Eysenbach* (2001) beschriebene Rolle mit aktiver Beteiligung am Behandlungsprozess (siehe dazu auch die Ausführungen in Abschnitt 4.6.1). Damit sind für Patienten entsprechende Informationssysteme zur Verfügung zu stellen. Ein Ansatz dafür ist

in der vorliegenden Arbeit mit der Implementierung eines Patientenportals einschließlich Ankopplung an das AMD vorgestellt. Damit erhält der Patient die Möglichkeit, in seine virtuelle Patientenakte Einsicht zu nehmen sowie eigene Daten einzutragen. Weiterhin erlauben Software-Agenten die Unterstützung von Patienten im Sinne der ambienten Intelligenz (siehe dazu die Ausführungen in Abschnitt 5.3.1). Grundlegende Arbeiten dazu werden z.B. bei *Moreno/Isern/Sánchez* (2003) dargestellt, die für eine Übertragung in das deutsche Gesundheitswesen entsprechend anzupassen sind.

Mit der ganzheitlichen Sicht auf einen Behandlungsprozess sind auch die Leistungserbringer mit einem adäquaten Informationssystem zu unterstützen. Das in der vorliegenden Arbeit beschriebene AMD erlaubt eine Orientierung an Prozessen sowie die Bereitstellung einer zusätzlichen Virtualisierungsschicht oberhalb bisheriger Informationssysteme. Damit kann es als Ergänzung zu den bisherigen, spezialisierten Informationssystemen betrachtet werden.

7.4 Labortests

Als weitere Evaluierungsmethode zur Bewertung der technologischen Eigenschaften werden in diesem Abschnitt Labortests beschrieben. Zur Qualitätssicherung von Software können unterschiedliche Ansätze mit jeweils variierender Aussagekraft über die erreichte Qualität eingesetzt werden (siehe Abbildung 7.4-1). *Menzies/Pecheur* (2005, 5-14) unterscheiden dabei die folgenden Abstufungen (siehe dazu auch Timm/Scholz/Fürstenau 2006, 533-537):

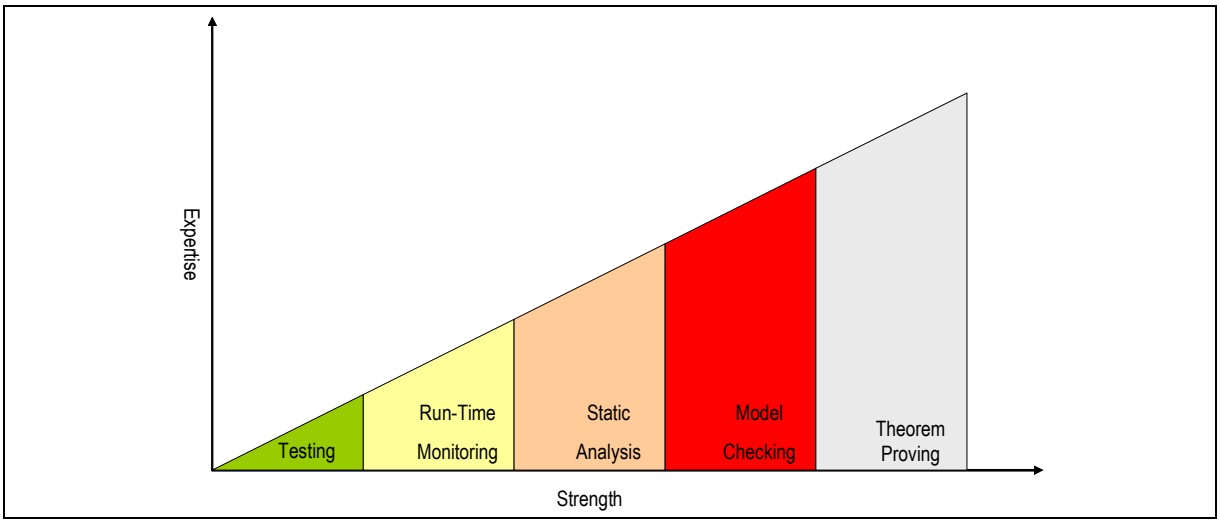

Abbildung 7.4-1: ***Qualitätssicherung für Systeme der KI***
Quelle: *Menzies/Pecheur* (2005, 5)

- *Testen*: Vergleich der Ausgaben des entwickelten Systems bei Eingabe von Werten über ein Testprogramm mit den spezifizierten Ergebnissen
- *Monitoring zur Laufzeit*: Überprüfung von Bedingungen durch Annotationen im Programmtext (z.B. in temporaler Logik) sowie automatisierte Analyse der generierten Log-Dateien
- *Statische Analyse*: Automatisierte Analyse des Codes ohne Ausführung des Programms

- *Model Checking* (Clarke/Emerson 1981): Überprüfung des gesamten Zustandsraumes eines Programms und Vergleich des Programmmodells mit der Spezifikation (oftmals über formale Logik)
- *Theorem-Beweis*: Rechnerunterstützte formale Verifikation der Korrektheit eines Programms mit Hilfe von mathematischen Regeln (Timm/Scholz/Fürstenau 2006, 536-537)

Entsprechend den Ausführungen bei *Timm/Scholz/Fürstenau* (2006, 541-542) kann das Unit Testing (White Box Test) geeignet durch Model Checking oder Theorem-Beweise (für die Repräsentation von Wissen und Inferenzen eines Agenten) sowie syntaktische Überprüfungen (für die Darstellung der Agenteninteraktionen als Zeichenkette entsprechend der Grammatik der Sprache über die Interaktionen) ergänzt werden, um eine Qualitätssicherung in einem agentenbasierten System zu etablieren. *Timm/Scholz/Fürstenau* (2006, 547) zeigen insbesondere auf, dass Ansätze des Model Checkings bei der Anwendung in Multiagentensystemen zwar zu unendlichen zu überprüfenden Zuständen führen können (Kacprzak/Lomuscio/Penczek 2004), aber dennoch über ein geeignetes Vorgehen (Kacprzak/Lomuscio/Penczek 2004) hinsichtlich ihrer Komplexität beherrschbar sind. Bezüglich Werkzeugunterstützung zur Qualitätssicherung von Agentensystemen kann jedoch festgestellt werden, dass diese ungenügend ist (Timm/Scholz/Fürstenau 2006, 548). Dies trifft insbesondere auf die Qualitätssicherung für Interaktionen zwischen Agenten zu (Timm/Scholz/Fürstenau 2006, 542). Wegen der genannten Problemstellungen werden in der vorliegenden Arbeit lediglich Unit Tests durchgeführt. Diese werden durch vergleichbare Verfahren für das Testen von Subsystemen und des Gesamtsystems ergänzt.

Das Ziel des Testens ist die Erhöhung der Software-Qualität (Timm/Scholz/Fürstenau 2006, 534; Thaller 2002). Dabei können lediglich vorhandene Fehler ggf. identifiziert werden. Eine Korrektheit der Programme kann damit aber nicht abgeleitet werden (Dijkstra 1969; Timm/Scholz/Fürstenau 2006, 534). Software-Qualität kann hinsichtlich folgender Aspekte differenziert werden (Riedemann 1997; Timm/Scholz/Fürstenau 2006, 532):

- Funktionalität
- Zuverlässigkeit
- Benutzerfreundlichkeit
- Effizienz
- Anpassbarkeit

Entlang dieser Ziele wird im Folgenden auf diese Qualitätsattribute für die vorliegende Arbeit Bezug genommen:

In Anlehnung an das etablierte Vorgehensmodell V-Modell® XT[98] (Broy/Rausch 2005), die Ausführungen bei *Timm/Scholz/Fürstenau* (2006, 537-538) bzw. *Thaller* (2002) sowie passend zu der in der vorliegenden Arbeit gewählten ArBaCon-Methode (siehe Abschnitt 1.7)

[98] V-Modell® ist eine geschützte Marke der Bundesrepublik Deutschland. Ausführliche Informationen zu dem Vorgehensmodell V-Modell XT können auf den folgenden Seiten abgerufen werden: http://www.kbst.bund.de/cln_046/Content/Home/homepage.html__nnn=true, zugegriffen am 12.07.2007.

wird entsprechend der jeweiligen Implementierungsphase ein geeignetes Testverfahren zur Überprüfung der Funktionalität ausgewählt (siehe Abbildung 7.4-2).

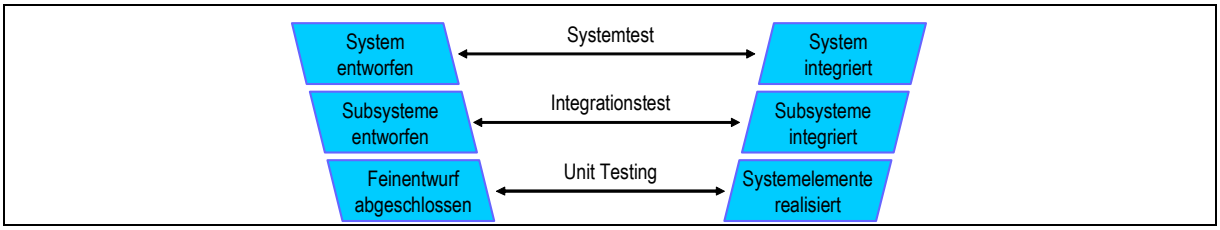

Abbildung 7.4-2: ***Testverfahren auf unterschiedlichen Abstraktionsebenen***
Quelle: Eigene Darstellung, in Anlehnung an *Bundesrepublik Deutschland* (2004, 28) und *Hoffmann/Leimeister/Krcmar* (2007, 144)

Für die Zwecke grundlegender Tests in der vorliegenden Arbeit ist die JADE-Erweiterung Test Suite[99] ausreichend. Damit können Unit Tests, d.h. die Überprüfung der Funktionalität von einzelnen Komponenten (Bruegge/Dutoit 2000, 344) bzw. Agenten, agentenbasiert durchgeführt werden. Diese Agenten entsprechen denjenigen, die in Etappe 3 der ArBaCon-Methode (siehe Abschnitt 1.7) implementiert wurden: Zur Qualitätssicherung wurden die in der vorliegenden Arbeit implementierten Agenten hinsichtlich ausgewählter Funktionalität durch Unit Tests getestet. Entsprechend der Idee des White Box Testings werden dabei die Interna der Agenten bzw. Komponenten überprüft. Dazu werden die Ausgaben der Agenten bzw. Komponenten als Reaktionen auf die Eingaben mit der Spezifikation verglichen. Entsprechend dem Aspekt des Evaluierungsfokus wird damit Qualitätssicherung auf der Mikroebene gewährleistet. Für ausgewählte nicht agentenbasierte Teile des Systems wird das etablierte Werkzeug JUnit[100] für Testzwecke auf der White Box-Ebene eingesetzt. Damit werden insbesondere Ausschnitte der GUI-, Persistenz- sowie Portalfunktionalitäten überprüft. Dazu ist das jeweilige Verhalten auf die Eingabe der Testwerte mit der Spezifikation zu vergleichen.

Ergänzend zu ausgewählten White Box-Testläufen werden die elementaren Komponenten der Subsysteme sukzessive integriert (Mesoebene zwischen Agenten- und Systemtests, Bruegge/Dutoit 2000, 353). Damit kann insbesondere die Interaktion zwischen den Komponenten bzw. Agenten getestet werden. Schließlich stehen am Ende dieses Testverfahrens überprüfte Subsysteme zur Verfügung. Diese entsprechen den Artefakten am Ende von Etappe 3 der ArBaCon-Methode (siehe Abschnitt 1.7), die das Ergebnis der Subsystemintegration darstellen. Bei der Qualitätssicherung auf diesem Abstraktionsniveau sind die Subsysteme gegen ihre jeweiligen Spezifikationen zu testen.

Abschließend werden auf Makroebene Systemtests durchgeführt, um die Qualität der Systemfunktionalität zu gewährleisten. Dabei werden insbesondere die funktionalen und nicht funktionalen Systemeigenschaften überprüft (Bruegge/Dutoit 2000, 358). Das Systemverhalten wird dabei gegen Anwendungsfälle getestet (Bruegge/Dutoit 2000, 359). Diese Anwendungsfälle

[99] JADE Test Suite einschließlich Dokumentation kann auf den folgenden Seiten abgerufen werden: http://jade.tilab.com/, zugegriffen am 09.02.2007.

[100] Weiterführende Informationen zu JUnit sowie die Testumgebung selbst stehen auf den folgenden Seiten zur Verfügung: http://www.junit.org/index.htm, zugegriffen am 09.02.2007.

(siehe Abschnitt 6.6.2) werden in der vorliegenden Arbeit in Etappe 1 der ArBaCon-Methode (siehe dazu Abschnitt 1.7) identifiziert. Mit diesen Ausführungen wurde auf unterschiedlich abstrahierter Systemebene gezeigt, wie die Software-Qualität hinsichtlich der Funktionalität in der vorliegenden Arbeit gewährleistet werden kann.

Für die Diskussion der Zuverlässigkeit in agentenbasierten Systemen wird auf die Ausführungen von *Nimis et al.* (2006) verwiesen: Da die Zuverlässigkeit des Prototyps in der vorliegenden Arbeit nicht im Fokus steht, wird im Folgenden auf entsprechende Ausführungen verzichtet. Da die Benutzeroberfläche von der implementierten Technologie abstrahiert und deshalb für die wissenschaftlichen Ergebnisse der vorliegenden Arbeit nicht von Bedeutung mit umfangreicher Tragweite ist, wird auch die Benutzerfreundlichkeit[101] nicht weiter betrachtet. Für die prototypische Realisierung und die in dieser Arbeit beschriebenen Ziele ist auch die Effizienz nicht von wesentlicher Bedeutung. Effizienzbetrachtungen werden z.B. bei *Cortese/Quarta/Vitaglione* (2002) und *Chmiel et al.* (2005) durchgeführt. Deshalb wird dieser Aspekt in der vorliegenden Arbeit nicht weiter betrachtet.

Die Anpassbarkeit des entwickelten Prototyps wurde bereits über die Realisierung unterschiedlicher Funktionalitäten und ihre Integration in das AMD demonstriert (siehe dazu die Abschnitte 6.7.3 mit 6.7.6 sowie 6.8.4 mit 6.8.8). Damit konnte die Modularität des AMD-Konzepts nachgewiesen werden. Weiterhin zeigten die Ausführungen in Abschnitt 3.4.2.7, dass sich agentenbasierte Systeme besonders durch ihre Flexibilität wegen ihrer losen Kopplung und ausgeprägten Modularisierung auszeichnen. Somit wird die Anpassbarkeit durch die vorliegende Implementierung umgesetzt.

Die Ausführungen in diesem Abschnitt zeigen, dass für die Qualitätssicherung von agentenbasierten Systemen noch besondere Problemstellungen zu lösen sind. Deshalb wird in der vorliegenden Arbeit nur mit dem Verfahren des Testens Qualitätssicherung erzielt. Dabei wurden insbesondere die Qualitätsaspekte der Funktionalität und Anpassbarkeit fokussiert, weil diese aus dem Anforderungskatalog aus Abschnitt 6.3.6 abgeleitet werden können. Als Ergebnis kann festgehalten werden, dass diese Anforderungen erfüllt werden können.

7.5 Relation zum Begründungszusammenhang

Für die folgenden Ausführungen dient Tabelle 1.9-2 als Grundlage. Dazu werden die dort identifizierten Kriterien in Relation mit den Ergebnissen der vorliegenden Arbeit gesetzt, um die wissenschaftliche Leistung herausarbeiten zu können.

Die Motivation bzw. Legitimation der implementierten Prototypen wird aus den Eigenschaften des Gesundheitswesens, ausführlich beschrieben in Abschnitt 4, abgeleitet. Dabei können insbesondere Defizite in der Informationslogistik identifiziert werden. Die Herausarbeitung der Besonderheiten der medizinischen Domäne bildet die Grundlage für die anschließende Eruierung von Anforderungen und deren Konsolidierung in einem Anforderungskatalog, womit auch die Angemessenheit von letzterem gegeben ist.

Die Spezifikation der Anforderungen an den Lösungsansatz bzw. die prototypische Implementierung wird mit der folgenden Argumentationslinie in der vorliegenden Arbeit begrün-

[101] Für weitere Ausführungen zur Benutzerfreundlichkeit wird z.B. auf die DIN-Norm 66234 oder *Shneiderman/Plaisant* (2005) verwiesen.

det: In Abschnitt 2 werden Agentensysteme als eine mögliche Erweiterung der Abstraktion in der Informatik eingeführt. Um aus elementaren Software-Elementen umfangreiche Anwendungen implementieren zu können, werden etablierte Architekturprinzipien eingesetzt. Dazu werden in Abschnitt 3 typische Architekturen insbesondere zur Umsetzung von verteilten Systemen sowie zu ihrer Integration beschrieben. In einer Evaluierung dieser Software-Architekturen kann gezeigt werden, dass sich agentenbasierte Systeme besonders durch ihre Flexibilität auszeichnen. Diese ist gegeben durch ihre ausgeprägte lose Kopplung und die Modularisierung. Dadurch können Software-Agenten flexibel auf Veränderungen in der Umwelt reagieren. Insgesamt wird die für Integrationsszenarien erforderliche Flexibilität vorteilhaft in agentenbasierten Systemen unterstützt.

Als Vorbereitung für eine Abbildung des Anforderungskatalogs durch geeignete Konstrukte der Software-Technik wird das Gesundheitswesen als CAS betrachtet (Abschnitt 5.1), welches sich durch ein MAS adäquat abbilden lässt (Abschnitt 5.3.1). Der in Abschnitt 4 aus den Eigenschaften der Domäne Gesundheitswesen abgeleitete Anforderungskatalog wird in Abschnitt 5.4 in seiner Realisierung beschrieben, wobei die Potenziale von Software-Agenten berücksichtigt werden. Die Betrachtung von anderen agentenbasierten Arbeiten (siehe Abschnitt 5.5) zeigt, dass diese hinsichtlich der Lösung von Problemstellungen im Gesundheitswesen ausbaufähig sind. Zusätzlich kann aus der Vorteilhaftigkeit der integrierten Versorgung (Abschnitt 4.2.7) sowie die durch die TI für die eGK gegebenen Potenziale die Nachhaltigkeit des in der vorliegenden Arbeit entwickelten Konzepts abgeleitet werden. Abschließend wird der Anforderungskatalog in Abschnitt 6.3.6 konkretisiert, indem ausgewählte Rahmenbedingungen aus der Domäne bzw. technische Gegebenheiten in einem Anwendungsfall berücksichtigt werden.

Die wesentlichen Inhalte des Lösungskonzepts und seiner Umsetzung wurden in den Abschnitten 6.5 mit 6.8 mit ihren jeweiligen Entwurfs- bzw. Implementierungsentscheidungen dargestellt. Ein wesentliches Ergebnis ist dabei die Darstellung des Lösungsansatzes der virtuellen Patientenakte auf der Basis des AMD-Konstrukts, womit eine zusätzliche Virtualisierungsschicht oberhalb der Ebene der bisherigen Informationssysteme zur Informationsintegration im Gesundheitswesen gebildet werden kann.

Zur Überprüfung der vorliegenden Arbeit als wissenschaftliche Leistung werden die folgenden Zusammenhänge beschrieben: Die Originalität wird aus den in Abschnitt 5.5 beschriebenen verwandten Arbeiten abgeleitet. Dabei wird insbesondere aufgezeigt, welche Defizite jeweils identifiziert werden können, womit auch das Alleinstellungsmerkmal der vorliegenden Arbeit herausgearbeitet ist. Diese Arbeiten können zusätzlich als Begründung dafür herangezogen werden, dass sich agentenbasierte Systeme geeignet im Gesundheitswesen einsetzen lassen. Der Generalisierungsanspruch der vorliegenden Arbeit wird durch die beschriebenen Erweiterungen (Einbettung in die TI, Portierung auf mobile Endgeräte, Integration von Scheduling- und Prozesssteuerungskomponenten, Ankopplung an ein Patientenportal) des AMD erreicht. Dabei ist insbesondere die Übertragbarkeit auf andere medizinische Einrichtungen als auf diejenige gegeben, die im Anwendungsfall beschrieben wurde. Eine weitergehende Generalisierung wird in Abschnitt 7.13.1 beschrieben, indem dort Entwicklungsperspektiven für ein Informationssystem im Gesundheitswesen dargestellt werden.

Der Vergleich der vorgeschlagenen Lösung mit dem Kriterienkatalog wurde bereits in den Abschnitten 7.1 mit 7.4 durch die Evaluierung bzw. Labortests gezeigt. Die Implementierung wurde insbesondere auf der Basis eines ausgewählten Behandlungsprozesses in einem Referenzklinikum umgesetzt und damit in ein Praxisszenario eingebettet. Durch den Bezug zu diesem Behandlungspfad ergibt sich insbesondere auch das Alleinstellungsmerkmal der vorliegenden Arbeit.

Die weiteren Aspekte aus Tabelle 1.9-2 werden in den folgenden Abschnitten nach der Bewertung der implementierten Prototypen zur Darstellung potenzieller Mängel und ihrer Beseitigung beschrieben.

7.6 Bewertung des Prototyps zur Informationsintegration

Ziel des Prototyps ist die Erarbeitung einer prototypischen Implementierung zur Informationsintegration im Gesundheitswesen. Die Tragfähigkeit des Designs bzw. der Implementierung wurde bereits in Abschnitt 7.3 begründet. Im Folgenden wird eine Bewertung der erzielten Eigenschaften der Implementierung vorgenommen:

In der Realisierung wurden exemplarisch unterschiedliche Informationssysteme integriert. Dabei wurden jedoch Schnittstellen teilweise simuliert. Vollständige Schnittstellenfunktionalität kann im Fall von SAP IS-H bzw. i.s.h.med durch geeignete BAPI- bzw. ABAP-Erweiterungen ergänzt werden. Dennoch konnte die prinzipielle Anbindung an ein etabliertes KIS demonstriert werden. Auch die Integration eines PVS wurde in der vorliegenden Arbeit nur simuliert. Dies ist im Hinblick auf die Ziele des Prototypenbaus legitim (siehe dazu Abschnitt 1.8).

Die virtuelle Patientenakte wurde in der vorliegenden Arbeit als Mehrwertdienst innerhalb der TI für die eGK realisiert. Dazu wird exemplarisch auf der eGK eine Verweisliste zu Speicherorten von medizinischen Daten des Patienten abgelegt. Alternativ hätte die Verweisliste auch auf einem Server der TI gespeichert werden können. Da der Konnektor den Zugriff auf die Verweislisten vor den Primärsystemen verbirgt, ist die Anpassbarkeit der Implementierung gegeben, um die tatsächlichen Anforderungen der TI zu erfüllen. Insgesamt trägt die eGK damit zur Konstruktion einer virtuellen Patientenakte bei, die alle bisherigen Informationen zu einem Patienten aggregiert. Bei der Implementierung wurde dabei auf eine detaillierte Umsetzung der Anforderungen aus der TI verzichtet. Dazu wird bei den Interaktionen der Konnektoren in Richtung TI von der Kommunikationsinfrastruktur abstrahiert. Für eine weitere Berücksichtigung der Anforderungen der TI ist eine engere Ankopplung der entwickelten Lösung erforderlich.

Obwohl die TI mit einer Pull-Architektureigenschaft angestrebt wird, wurde in der vorliegenden Arbeit eine Realisierung der virtuellen Patientenakte über eine Push-Funktionalität vorgeschlagen. Damit lassen sich wesentliche Nachteile der Pull-Funktionalität beseitigen.

Das Konzept des AMD erlaubt bei der in der vorliegenden Arbeit beschriebenen Implementierung lediglich eine Zusammenstellung der medizinischen Informationen zur Laufzeit. Ein persistentes Ablegen der aggregierten Daten ist nicht umgesetzt, im Design der Implemen-

tierung (siehe dazu die Ausführungen zu SaveTempRecord in den Abschnitten 6.6.4.3 und 6.8.1) jedoch berücksichtigt. Aus Performanzgründen ist eine persistente Speicherung von Daten sinnvoll, um nicht bei jeder Einsichtnahme in die virtuelle Patientenakte den umfangreichen Abfrageprozess erneut zu durchlaufen und auch bei fehlender Konnektivität zumindest die seit der letzten Aggregation verfügbaren Dateien einsehen zu können. Die persistente Speicherung dieser Daten setzt jedoch geeignete Synchronisationsmechanismen voraus, durch welche die Daten für den Leistungserbringer in ihrer aktuellen Version bereitgestellt werden.

Der bei *Zachewitz/Schwolow* (2004) und *Zachewitz* (2004a; 2004b) beschriebene Mechanismus zur Konsistenzsicherung von Informationen zwischen Agenten kann dabei in einer geeigneten Anpassung auch für ein AMD eingesetzt werden. Der Konsistenzsicherungsmechanismus von *Zachewitz* (2004a; 2004b) kann als eine mögliche Erweiterung der AMD-Implementierung betrachtet werden, weil dort eine entsprechende Erweiterung der FIPA-Architektur vorgeschlagen wird und in der vorliegenden Arbeit ebenso diese Agentenplattform eingesetzt wurde. Die grundlegende Idee der Konsistenzsicherung bei *Zachewitz/Schwolow* (2004) und *Zachewitz* (2004a; 2004b) entspricht dem bei *Gamma et al.* (1995, 293-303) beschriebenen Observer-Entwurfsmuster: Dabei registrieren sich Observer-Objekte bei Subjekten. Sobald eine Änderung am Datenmodell des Subjektes vorgenommen wird, werden vom Subjekt alle bei ihm registrierten Observer-Objekte registriert. Diese reagieren entsprechend auf diese Änderungen. Analog dazu registrieren sich (attach-Nachricht bei ‚Gamma et al. 1995, 294) Agenten mit der Aufgabe der Datenkapselung bei der zentralen Wartungskomponente im Vorschlag von *Zachewitz* (2004a). Sobald in einem Agenten eine Datenänderung vorgenommen wird, wird diese Komponente entsprechend benachrichtigt (notify-Nachricht bei ‚Gamma et al. 1995, 294-295). Von der Wartungskomponente werden dedizierte Wartungsagenten beauftragt, die Wartung (update-Nachricht bei ‚Gamma et al. 1995, 294-295) bei den registrierten Agenten durchzuführen. Da bei der in der vorliegenden Arbeit beschriebenen Implementierung zur Kapselung von veränderlichen Daten Wrapper-Agenten der Informationssysteme eingesetzt werden, müssten sich diese und AMD-Agenten bei der Wartungskomponente registrieren. Werden Daten in ein Informationssystem eingetragen oder verändert, werden die Datenbestände der entsprechenden AMD-Agenten über die Wartungskomponente bzw. die Wartungsagenten aktualisiert. Mit diesem Interaktionsmuster ist die Kommunikation zwischen Agenten zur Aktualisierung von Daten nur dann erforderlich, wenn tatsächlich Änderungen vorliegen. Somit kann die Performanz bei der Datenaggregation gesteigert werden.

Demonstriert werden konnten in der Implementierung die Nutzenstiftung sowie die technische Realisierbarkeit der Unterstützung von ausgewählten etablierten medizinischen Standards. Für eine Untermauerung der erzielten Ergebnisse sind jedoch weitere Standards in die Implementierung zu integrieren, um die durchgängigen Fähigkeiten von Software-Agenten als Informations-Broker zu demonstrieren.

Die implementierte Informationsaggregation von medizinischen und administrativen Daten eines Patienten berücksichtigt keine erforderlichen gesetzlichen Restriktionen zur Sicherstellung z.B. des Datenschutzes. Mit dem AMD wurde vielmehr ein prinzipieller Lösungsvorschlag erarbeitet, der für einen Einsatz in der Praxis derart anzupassen ist, dass die nötigen Anforderungen zur Gewährleistung von Sicherheitseigenschaften vollständig erfüllt werden können.

7.7 Bewertung der Einbettung in die Telematikinfrastruktur

Ausgewählte Eigenschaften, die durch die TI vorgegeben sind, konnten in der prototypischen Implementierung umgesetzt werden (siehe dazu Abschnitt 6.8.3). Ein Anspruch der vollständigen Implementierung aller Spezifikationen der TI wird in der vorliegenden Arbeit nicht erhoben.

Obwohl keine konzeptionellen Hindernisse identifiziert werden konnten, steht die in JADE-S implementierte Funktionalität einer vollständigen Umsetzung der ausgewählten Anforderungen entgegen. Dabei konnten die folgenden Defizite erkannt werden:

- Fehlende technische Unterstützung von Verschlüsselung zwischen verschiedenen Plattformen
- Eingeschränkte Sicherheitseigenschaften bei mobilen Agenten
- Eingeschränkte Sicherheitseigenschaften bei der Unterstützung von mobilen Endgeräten

Diese Aspekte können aber nicht als inhärente Defizite von Agentensystemen, sondern vielmehr als Notwendigkeit für eine entsprechende Weiterentwicklung von JADE-S betrachtet werden. Weil fehlende Funktionalität von JADE-S zur Verfügung zu stellen ist, wird auf eine entsprechende eigene Erweiterung in der Implementierung verzichtet, um die Kompatibilität der Applikation mit der Weiterentwicklung der JADE-Erweiterungen zu gewährleisten.

7.8 Bewertung des mobilen Prototyps

Die prototypische Implementierung zur Aggregation von Patientendaten wurde in ihrer fast vollständigen Funktionalität[102] auf ein ausgewähltes mobiles Endgerät portiert. Obwohl die Ressourcen von mobilen Endgeräten prinzipiell beschränkt sind, kann die Performanz der mobilen Anwendung als ausreichend beurteilt werden (Hillebrand 2006, 84). Die entwickelte Anwendung kann entsprechend ihrer Plattformunabhängigkeit, induziert durch die Java-Implementierung, auch auf anderen mobilen Endgeräten eingesetzt werden, für die die gewählte Laufzeitumgebung geeignet konfiguriert werden kann (Hillebrand 2006, 87-88). Weiterhin abstrahiert die für die Anwendung gewählte Architektur von der zugrunde liegenden Netzwerkverbindung. Dementsprechend können je nach Verfügbarkeit auf dem mobilen Endgerät unterschiedliche Funktechnologien verwendet werden, auf deren Basis sich eine TCP/IP-Verbindung einrichten lässt (für mobile Geräte üblicherweise Bluetooth, WLAN, GPRS oder UMTS, Hillebrand 2006, 108). Aus technischer Hinsicht ist dabei jedoch unklar, in welchem zeitlichen Umfang Abbrüche der Funkverbindung maskiert werden können. Deshalb besteht der Bedarf für entsprechende Explorationen, um letztendlich die Einsatzfähigkeit auf der Basis der aktuellen Ausbaustufe bzw. die Anforderung einer Anpassung der JADE-LEAP-Plattform zu eruieren.

Die mit Java Swing zur Verfügung gestellte Funktionalität, HTML-Dokumente, die in der Desktop-Anwendung aus den transferierten HL7 CDA-Dokumenten generiert werden, direkt

[102] Da die technische Realisierung der Einbettung von multimedialen Daten in die virtuelle Patientenakte für die Desktop-Applikation bereits demonstriert wurde, wurde auf eine Portierung dieser Funktionalität auf die mobile Anwendung verzichtet, weil dadurch keine weiteren wesentlichen Erkenntnisse erreicht würden.

in Elemente der Benutzeroberfläche zu integrieren, ist in der auf mobilen Endgeräten vorhandenen Bibliothek Java AWT nicht gegeben. In der vorliegenden Implementierung werden die empfangenen Dokumente entsprechend der ex ante bekannten Struktur geparst und an der Benutzeroberfläche dargestellt. Beim Parsen des Dokuments werden Java AWT-Komponenten entsprechend den Elementen des XML-Dokuments generiert. Die Transformation von HL7-CDA nach HTML über XSLT und eine direkte Visualisierung in einer GUI-Komponente könnte auch auf mobilen Endgeräten realisiert werden. Dazu müssten geeignete Bibliotheken implementiert werden, welche HTML-Dokumente zur Anzeige bringen können.

Zur Gewährleistung der Sicherheit der drahtlos übertragenen Daten wurden in der mobilen Implementierung keine Maßnahmen ergriffen, weil diese in der vorliegenden Arbeit nicht fokussiert werden. Um diese Defizite zu beheben, werden bei *Hillebrand* folgende Schritte vorgeschlagen (2006, 89-90), wobei unterschiedliche Schichten differenziert werden:

- *Hardware*
  - Verschlüsselung nach Wi-Fi® Protected Access (WPA™)
  - Zulassung nur vorgegebener MAC-Adressen (Media Access Control) innerhalb des Funknetzwerkes
  - Vergabe von statischen IP-Adressen und Verzicht auf dynamische Adressen durch Dynamic Host Configuration Protocol (DHCP)
- *Netzwerkprotokolle*: Gesicherte Verbindungen durch den Einsatz von Transport Layer Security (TLS)
- *Anwendungsschicht*: Eine Verschlüsselung von ACL-Nachrichten mit JADE-S ist – entgegen den Ausführungen in Abschnitt 6.3.4, nach denen die Sicherheitseigenschaften nur auf MIDP-Geräten nicht zur Verfügung stehen – in der aktuellen Version im Split-Ausführungsmodus auf der Basis der Konfiguration J2ME/CDC/Personal Profile nicht möglich. Für eine Verschlüsselung sind entsprechende Erweiterungen auf der Basis von Verschlüsselungsbibliotheken[103] zu ergänzen. In der vorliegenden Arbeit wurde jedoch aus Kompatibilitätsgründen auf eine Erweiterung von Sicherheitseigenschaften als Ergänzung zu JADE-S verzichtet.

Mit der Portierung der ausgewählten Funktionalität der Informationsaggregation auf ein mobiles Endgerät konnte die prinzipielle Realisierung von agentenbasierten Anwendungen auf mobilen Endgeräten demonstriert werden, womit ein Zielaspekt der vorliegenden Arbeit erreicht wurde. Die Portierung der übrigen, nur in der Desktop-Applikation implementierten Funktionalitäten würde jedoch einen zusätzlichen Mehrwert schaffen, weil mobile Geräte das informationslogistische Prinzip hinsichtlich des Entstehungsortes von Informationsbedarf vorteilhaft unterstützen können:

[103] Bspw. implementiert die Bibliothek Bouncy Castle Crypto API geeignete Funktionalitäten für die Laufzeitumgebungen J2ME und J2SE. Weitere Informationen zur dieser API können auf den folgenden Seiten abgerufen werden: http://www.bouncycastle.org/, zugegriffen am 13.02.2007. Alternativ könnte eine entsprechende Anpassung der Funktionalität von JADE, JADE-LEAP bzw. JADE-S eine geeignete Basis bilden, um Sicherheitseigenschaften auch für mobile Anwendungen zur Verfügung zu stellen. Zur Sicherung der Kompatibilität der JADE-Erweiterungen ist diese Variante zu präferieren.

- Die Ankopplung an ein Patientenportal auch über das mobile Endgerät erlaubt dem Patienten einen ortsunabhängigen Zugriff auf seine Daten. Der Zugriff auf das in der vorliegenden Arbeit gewählte Liferay-Portal durch ein mobiles Endgerät ist prinzipiell gegeben (siehe dazu die Ausführungen in Abschnitt 6.7.5.1).
- Die Vereinbarung von Terminen kann mit dem mobilen Endgerät am Entstehungsort des Informationsbedarfs und von Daten durchgeführt werden. Damit werden Medienbrüche reduziert.
- Die Erweiterung der Prozessunterstützung auf mobile Endgeräte würde es dem medizinischen Personal erlauben, entsprechend dem informationslogistischen Prinzip Informationen abzufragen. Dies erhöht die Arbeitseffektivität und -effizienz.

7.9 Bewertung der Scheduling-Applikation

Mit der Implementierung der Scheduling-Anwendung konnte demonstriert werden, dass sich agentenbasierte Systeme insbesondere für die durch hohe Dynamik gekennzeichnete Domäne Gesundheitswesen eignen. Dazu können in dem implementierten Scheduling-Algorithmus insbesondere Änderungen im Gesundheitszustand von Patienten berücksichtigt werden. Diese führen dazu, dass Patienten mit höherer Priorität, also z.B. in einem Notfall, bei der Terminplanung bevorzugt werden. In der Implementierung wurde dazu eine geeignete Kostenfunktion für Patienten realisiert.

Auf die Übertragung dieser Kostenfunktion auf die Ressourcen-Agenten wurde in der vorliegenden Arbeit verzichtet, weil damit kein weiterer Mehrwert zur Demonstration der prinzipiellen Tragfähigkeit der Architektur erreicht würde. Zur Beseitigung dieses Defizits würde die Übertragung der analog bereits implementierten Funktionalität auf Ressourcen-Agenten ausreichen.

Außerdem wurde in der vorliegenden Arbeit die Integration der Terminpläne des für die Untersuchungen erforderlichen medizinischen Personals mit seinen jeweiligen Präferenzen für die Bildung von Behandlungsteams sowie der benötigten Geräte nicht berücksichtigt. Präferenzen werden im Scheduling-Kontext bei *Becker/Czap* (2006) betrachtet. Deshalb kann in der vorliegenden Arbeit auf diesen Aspekt verzichtet werden. Auf die explizite Berücksichtigung von für die Behandlungen erforderlichen Geräten kann in der vorliegenden Arbeit verzichtet werden, weil diese als Ressourcen modelliert werden können, die in der Implementierung bereits dargestellt wurden. Somit beschränkt sich die Integration von Geräten auf die Gewährleistung einer ausreichenden Skalierbarkeit der Agentenplattform.

Insgesamt kann festgestellt werden, dass mit der vorliegenden Scheduling-Applikation eine geeignete Basis bereitgestellt wird, auf der weitere Funktionalitäten umgesetzt werden können. Weitere Ausbaupotenziale als diejenigen, die in diesem Abschnitt bereits beschrieben wurden, sowie mögliche Lösungen werden auch in Abschnitt 6.7.4.3 dargestellt.

7.10 Bewertung der Kopplung des Agentensystems mit einem Portal

In der prototypischen Implementierung konnte demonstriert werden, dass eine Anbindung des AMD an ein Portalsystem über Web Services implementiert werden kann. Dazu konnte die JADE-Erweiterung WSIG (siehe Abschnitt 6.3.5) erfolgreich als Gateway zwischen dem AMD und dem Portalsystem eingesetzt werden. Damit wird die angestrebte Patientenorientie-

rung erreicht, denn Patienten können nicht nur ihre eigenen Daten einsehen, sondern möglicherweise auch zusätzliche Informationen z.B. zur sportlichen Betätigung eingeben. Die Implementierung genügt daher dem von *Waegemann* (1999, 116) vorgeschlagenen Anspruch „Electronic Health Record“ in einer prototypischen Umsetzung. Dennoch bleiben die folgenden Aspekte in der beschriebenen Implementierung offen, können aber in weiteren Arbeiten entsprechend gelöst werden:

- *Vereinfachte Anbindung an das AMD*: Mit der unmittelbaren Interaktion des Portals mit dem AMD konnte demonstriert werden, dass das entwickelte Agentensystem mit dem Portal aus technischer Perspektive gekoppelt werden kann. Dabei wurde jedoch vernachlässigt, dass in einem realitätsnahen Szenario zunächst die Patientenakte zu öffnen wäre, bevor an diese eine Anfrage gestellt werden könnte. Da diese Funktionalität bereits für die Desktop-Applikation implementiert wurde (siehe Abschnitt 6.8.1), wurde zur Reduktion der Implementierungskomplexität auf eine erneute Umsetzung für die Portalanwendung verzichtet. Eine Anpassung der Implementierung würde entsprechend der Konstruktion der Applikation für die Dokumentenaggregation der Desktop-Anwendung erfolgen, bei der zunächst die Patientenakte zu öffnen ist. Damit werden aber keine weiteren Erkenntnisse erwartet, weshalb die beschriebene Einschränkung die Qualität der Ergebnisse der vorliegenden Arbeit nicht reduziert.
- *Fehlende Integration der Datenbank für das Portalsystem in die Architektur der TI*: Der Zugriff auf die Datenbank mit den Daten des Patienten, welche von ihm in das Portalsystem selbst eingegeben werden, müsste in der Lösungsarchitektur der TI als zusätzlicher Anwendungsdienst realisiert werden. Auch für eine Anbindung des AMD an das Portal über die Dienste der TI müsste ein weiterer Anwendungsdienst eingeführt werden, der als zentraler Dienst die Bereitstellung der Daten für das Portalsystem realisiert. Bei der in der vorliegenden Arbeit beschriebenen Implementierung wurde jedoch auf diese Umsetzung verzichtet, weil damit zusätzlicher Implementierungsaufwand entstanden wäre, der jedoch nicht zu weiteren Erkenntnissen hinsichtlich der Ankopplung eines AMD an ein Portalsystem geführt hätte.
- *Anzeige von Dokumenten*: Da die Kommunikation zwischen dem Portal und der WSIG-Erweiterung asynchron gestaltet ist, wurde eine Synchronisationsschicht integriert, die jedoch nur das erste der ankommenden Dokumente bei der Informationsaggregation verarbeitet. Ein vollständig implementierter Mechanismus mit dieser Funktionalität würde jedoch keinen weiteren Erkenntnisfortschritt induzieren, weshalb in der vorliegenden Arbeit darauf verzichtet wurde. Zur Beseitigung dieses Mangels könnte der Synchronisationsmechanismus geeignet erweitert werden, um alle ankommenden Dokumente bei der Informationsaggregation verarbeiten zu können.
- *Fehlende Integration der Daten aus dem Portal in das AMD*: In der vorliegenden Implementierung wurde die Ankopplung des AMD an das Portal nur unidirektional umgesetzt: Damit werden Informationen aus den verteilten Informationssystemen aggregiert und im Portal visualisiert. Die bidirektionale Anbindung würde die Aggregation von Daten im AMD auch aus dem Portalsystem erfordern. Dadurch könnten von den Leistungserbringern auch die von den Patienten selbst eingetragenen Daten eingesehen und in der weiteren Entscheidungsfindung berücksichtigt werden. Aus technischer Hinsicht hätte diese Funktionalität jedoch keinen weiteren Mehrwert beigetragen. Denn für die Realisierung wäre lediglich ein zusätzlicher Wrapper-Agent zu implementieren, der entsprechende ACL-Aufrufe in SQL-Abfragen transformiert und an die Datenbank weiterleitet. Die Ergebnisse der Datenbankabfrage würden von dem Wrapper-Agent in geeig-

nete ACL-Inhalte überführt werden. Dies wäre somit eine zusätzliche Erweiterung der bereits angebundenen Informationssysteme und würde nicht zu weiteren wissenschaftlichen Erkenntnissen beitragen.

- *Fehlende Berücksichtigung von Sicherheitseigenschaften*: Sicherheitseigenschaften wurden in der beschriebenen Implementierung nicht umgesetzt. Um die gewünschten Sicherheitseigenschaften der Vertraulichkeit, Authentifizierung, Integrität und Nichtabstreitbarkeit (Schneier 1996, 2) auch beim Einsatz von Web Services, realisiert durch SOAP-Aufrufe, zu gewährleisten, können in einer geeigneten Erweiterung die folgenden Mechanismen eingesetzt werden (Jeckle/Zengler 2002):
  - Integration einer für die SOAP-Anwendung weitgehend transparenten SSL-Schicht auf Protokollebene
  - XML-Signaturen (Bartel et al. 2002) und XML-Verschlüsselung (Imamura/Dillaway/Simon 2002) auf XML-Ebene

Die Ausführungen in diesem Abschnitt zeigen, dass bei der Anbindung eines Portals an das AMD Mängel identifiziert werden können. Letztere sind jedoch technischer Natur und können deshalb durch geeignete Maßnahmen beseitigt werden.

Zusammenfassend kann festgestellt werden, dass das in der vorliegenden Arbeit beschriebene AMD-Konzept die Ankopplung eines Patientenportals auf der Basis etablierter Standards unterstützt. Damit kann der Patient über die Einsichtnahme in seine Gesundheitsdaten und das Eintragen weiterer Informationen in den Behandlungsprozess integriert werden.

7.11 Bewertung der Prozessunterstützung

Mit einem exemplarischen Musterprozess konnte demonstriert werden, wie die Implementierung einer agentenbasierten Prozesssteuerung realisiert werden kann. Ausgangspunkt für letztere ist die Modellierung eines Prozesses in der AML-Beschreibungssprache. Diese Spezifikation wird, wie in den Abschnitten 6.7.6 sowie 6.8.7 beschrieben, transformiert, um schließlich als Eingabe für die Steuerungskomponente zu fungieren. Für die Abarbeitung eines solchen Prozesses zur Laufzeit wurden Behaviours zur Interaktion mit dem Benutzer, zur Bereitstellung von Informationen für den Benutzer und zur Integration anderer Systeme (Bergmann/Klinger/Winkler 2006, 76) implementiert. Dabei wurde insbesondere gezeigt, wie die Aggregation der Dokumente in einer virtuellen Patientenakte mit der umgesetzten Prozessunterstützung gekoppelt werden kann und die darin enthaltenen Informationen proaktiv zur Verfügung gestellt werden können. Somit wurde das AMD um eine Prozesssteuerungskomponente erweitert, die den aktiven Teil von aktiven medizinischen Dokumenten darstellt.

Trotz der Zielerreichung einer agentenbasierten Prozessunterstützung ist letztere nicht in allen Bereichen agentenbasiert umgesetzt. So wurde bei der Persistenzsicherung von Prozessen aus folgenden Gründen auf eine agentenorientierte Implementierung verzichtet (siehe auch Bergmann/Klinger/Winkler 2006, 66): Die Persistenzschicht könnte zwar grundsätzlich über das etablierte Wrapper-Muster (Jennings 2001, 40) an das Agentensystem angebunden werden. Daraus würden sich aber folgende technisch bedingte Nachteile ergeben:

- Für die Klassen ProcessState, Repository, ConstraintSet und Constraint, die den Zustand für einen Prozess bestimmen, hätten jeweils umfangreiche Ontologien[104] entwickelt werden müssen (Bergmann/Klinger/Winkler 2006, 66).
- Wie in Abschnitt 6.8.7 ausgeführt, erfolgt die Persistenzsicherung – entsprechend dem Transaktionsprinzip – nach jedem erfolgreich ausgeführten Prozessschritt. In einer agentenbasierten Umsetzung der Persistenzfunktionalität würde ein entsprechender Kommunikationsaufwand zwischen den Agenten entstehen, der wiederum zu Performanzverlust führen würde (Bergmann/Klinger/Winkler 2006, 66).

Weitere, bei *Bergmann/Klinger/Winkler* (2006, 87-90) detailliert beschriebene Grenzen der Implementierung der agentenbasierten Prozessunterstützung werden im Folgenden zusammengefasst:

- Die Prozesssteuerungskomponente bei *Meyer* (2004) erwartet als Eingabe XML-Dateien. Deshalb wird in der vorliegenden Implementierung aus den Datenbankeinträgen zur Laufzeit die Prozessverarbeitungsstruktur in einer Datei mit einem geeigneten Format abgelegt. Dieser Schritt würde obsolet, wenn die Prozesssteuerungskomponente derart erweitert würde, dass die Prozessinformationen unmittelbar aus der Datenbank ausgelesen werden.
- Die in dem PML-Element agent angegebene Bezeichnung spezifiziert einen Agenten, der das entsprechende Behaviour ausführen soll. Diese Funktionalität wird in der bei *Meyer* (2004) implementierten Prozesssteuerung nicht unterstützt. Deshalb wurde auch in der Prozessunterstützung der vorliegenden Arbeit auf diese Funktionalität vorerst verzichtet. In der beschriebenen Implementierung werden die Behaviours deshalb alle von einem einzigen Agenten, dem ProcessMasterAgent, abgearbeitet. Um bei einer Umsetzung der Abarbeitung der Behaviours durch individuelle Agenten die verfügbaren Ressourcen überlegt einzusetzen, sollten beim Start der Prozesssteuerung nicht alle evtl. benötigten Agenten gestartet werden. Vielmehr sind die jeweiligen Agenten erst dann zu initialisieren, wenn die entsprechenden Vorbedingungen für das auszuführende Behaviour erfüllt sind. Dazu wäre eine geeignete Verwaltungskomponente im ProcessMasterAgent erforderlich.
- Obwohl in einer EPK Kreise syntaktisch erlaubt sind, wurde diese Modellierung in der Prozesssteuerung der vorliegenden Arbeit nicht umgesetzt. Die Ursache liegt in der Implementierung der Verarbeitungskomponente für Prozesse bei *Meyer* (2004): Da dort abgearbeitete Behaviours verworfen werden, müsste in der Prozesssteuerung eine geeignete Funktionalität implementiert werden, die eine erneute Instanzbildung von Behaviours sowie eine entsprechende Belegung ihrer Vorbedingungen ermöglicht, ohne jedoch das soeben ausgeführte Behaviour sofort erneut abzuarbeiten, da die Vorbedingungen dafür bereits erfüllt waren.
- Parallele Prozessschritte werden nicht implementiert. Dazu müsste das Behaviour ProcessorBehaviour als eine Unterklasse analog zur Klasse ParallelBehaviour des JADE-Rahmenwerks implementiert werden.

[104] Eine automatisierte Abbildung der Hibernate-Persistenzvorschrift auf geeignete Ontologien würde die Implementierungsarbeit durch die Reduktion des Aufwandes einerseits vereinfachen und andererseits potenzielle Fehler reduzieren, die durch eine manuelle Abbildung entstehen würden. In weiteren Arbeiten ist deshalb zu eruieren, wie eine solche Abbildung durch Werkzeuge geeignet unterstützt werden kann.

- Zur Unterstützung von Dynamik bei Prozessen ist in der Implementierung lediglich die Funktionalität zum Beenden und zur Neuzuweisung von Prozessen gegeben. Die bisher implementierte Lösung ist deshalb im Wesentlichen auf statische Prozesse beschränkt. Dies bedeutet, dass vorab durch einen Domänenexperten Prozesse einschließlich ihrer möglichen Ausnahmefälle definiert werden müssten. Damit wird der Aufwand für die Erarbeitung der Behandlungsprozesse sehr umfangreich. Gleichzeitig steigt der Aufwand für die Erstellung einer geeigneten Mapping-Datei, die die EPK-Funktionen den jeweiligen Agenten bzw. ihren Behaviours zuordnet. Um dynamische und ex ante nicht determinierbare Prozesse, wie sie im Gesundheitswesen häufig auftreten, unterstützen zu können, wird in der umgesetzten Implementierung die Grundlage für eine entsprechende Erweiterung gelegt: Können statt einem nachfolgenden Prozessschritt auch vorhergehende zu ihrer Abarbeitung ausgewählt werden, weil z.B. eine Operation aus gesundheitlichen Gründen nicht möglich ist oder ein Notfallpatient vorgezogen werden muss, können dynamisch auftretende Ereignisse in der Prozesssteuerung berücksichtigt werden. Außerdem wäre der dynamische Einsprung in andere besser geeignete Prozesse zu unterstützen. Weiterhin müsste in einem Behandlungspfad entschieden werden können, dass bestimmte Schritte übergangen werden. Für die beschriebenen Fälle wäre zunächst das Setzen der Vorbedingungen des nächsten auszuführenden Prozessschrittes erforderlich. Damit könnte die Prozesssteuerung das entsprechende Behaviour des Agenten ProcessMasterAgent starten. Bei der Wiederaufnahme eines bereits abgearbeiteten Prozessschrittes ist dieses Vorgehen allein aber nicht ausreichend, weil die Vorbedingungen der bereits ausgeführten Prozessschritte, die sich zwischen dem letzten ausgeführten Prozessschritt und dem nun erneut auszuführenden Schritt befinden, jeweils bereits erfüllt sind. Um diese Problematik zu umgehen, müsste für jeden ausgeführten Prozessschritt der jeweilige Zustand persistent abgelegt werden. So könnte nach Auswahl des Benutzers an denjenigen Punkt zurückgesprungen werden, der für die vorliegende Situation gewählt wird. Der gespeicherte Prozesszustand ermöglicht somit die Wiederaufnahme der Prozesssteuerung an der gewählten Stelle.
- XML-Editoren zur Bearbeitung von XML-Dateien sind zwar etabliert, ein geeignetes Werkzeug, das den Anwender explizit bei der Erstellung der Mapping-Datei EPKProcessMappingXML unterstützt, fehlt, würde aber eine wesentliche Hilfestellung darstellen. Für die Entwicklung von geeigneten Abbildungsvorschriften zwischen AML und PML sowie die Konstruktion eines adäquaten Werkzeugs sind jedoch umfangreiche Arbeiten erforderlich. Die Implementierung einer solchen Werkzeugunterstützung liegt nicht im Fokus der vorliegenden Arbeit.

Neben den genannten Defiziten kann ein weiterer Mangel identifiziert werden: Die Funktionalität zur Abarbeitung von als EPKen modellierten Prozessen ist durch die inhärente Beschränkung der EPK-Modelleigenschaften eingeschränkt. So wird in einer EPK-Darstellung die Reihenfolge von parallel auszuführenden Teilprozessen nicht spezifiziert. Die zeitliche Abfolge von parallelen Prozessteilen ist deshalb auch in der implementierten Prozessunterstützung nicht bestimmt und deshalb nicht deterministisch, sondern abhängig von der in dem JADE-Rahmenwerk gewählten Verarbeitungsreihenfolge des Behaviours ProcessorBehaviour bzw. der vererbenden Oberklasse CompositeBehaviour. Da bisherige Erfahrungen mit der Modellierung von Prozessen als EPKen im medizinischen Bereich zeigen, dass diese für die Integration von Domänenexperten geeignet sind (Schweiger/Krcmar 2005), ist weniger eine vollständig andere Modellierungsmethode zu wählen, sondern vielmehr eine geeignete Werkzeugunterstützung zu entwickeln, die dem Domänenexperten eine Transformation der EPK-Darstellung in eine PML-Darstellung erlaubt. In diesem Schritt kann das originäre EPK-

Modell durch zusätzliche Bedingungen verfeinert werden, die die Abarbeitung von parallelen Teilprozessen in ihrer zeitlichen Abfolge spezifizieren.

Hinsichtlich Visualisierung des zu verarbeitenden Prozesses als EPK können nach *Slavtchev* (2007, 56-57) die folgenden Mängel festgestellt werden: Generische Layout-Algorithmen der verwendeten Bibliothek JGraph Layout Pro liefern für die Anordnung von EPKen nur unzureichende Ergebnisse. Deshalb ist in weiteren Arbeiten ein geeigneter Layout-Algorithmus zu entwerfen, der in die prototypische Implementierung integriert werden kann, um die besonderen Charakteristika einer EPK zu berücksichtigen. Weil nicht alle für die Visualisierung erforderlichen Informationen in der relationalen Datenbank abgelegt werden, kann bei der Wiederaufnahme der Abarbeitung eines Prozesses die Markierung des aktuell ausgeführten Prozessschrittes ggf. nicht vorgenommen werden. Dazu müssten alle relevanten Daten in der relationalen Datenbank gespeichert werden, um den aktuellen Prozessschritt eindeutig markieren zu können. Weiterhin lässt sich festhalten, dass wegen der fehlenden Unterstützung von Kreisen in einer EPK-Struktur durch die agentenbasierte Prozessunterstützung (siehe dazu die Ausführungen weiter oben in diesem Abschnitt) bei der Visualisierung des gewählten Prozesses zwei Ausführungsmodi differenziert werden müssen: Werden Kreise in einer EPK identifiziert, wird nur der Wurzelprozess ohne die aufgelösten Prozesswegweiser visualisiert. Die Fortschrittsmarkierung erfolgt dabei lediglich bei EPK-Funktionen und nicht bei Prozesswegweisern. Wird die Ausführungskomponente der Prozesssteuerung derart angepasst, dass auch Kreise in einer EPK abgearbeitet werden können, ist eine entsprechende Änderung in der Visualisierungskomponente möglich (Slavtchev 2007, 57). Eine kreisfreie EPK mit Prozesswegweisern hingegen kann vollständig entfaltet werden und in ihren jeweiligen Verarbeitungszuständen visualisiert werden. Die Fortschrittsmarkierung erfolgt dabei in der vollständig dereferenzierten EPK-Darstellung.

Mit der prototypischen Implementierung der Prozessunterstützung ist zusammenfassend eine geeignete Grundlage erarbeitet, die entsprechend weiteren Anforderungen oder den identifizierten Mängeln angepasst werden kann. Damit kann das Ziel einer prototypischen Implementierung als erreicht betrachtet werden.

7.12 Gesamtbewertung

Hinsichtlich der Gesamtbewertung der prototypischen Implementierung werden die zwei Perspektiven technische Aspekte und Gegebenheiten im gewählten Referenzklinikum im Folgenden betrachtet:

Aus technischer Sicht erwies sich die verwendete Agentenplattform JADE als stabil genug, um auf dieser Basis die prototypische Implementierung sowohl auf Desktop- als auch auf mobilen Endgeräten zu konstruieren. Insbesondere ist dabei die FIPA-Kompatibilität vorteilhaft, die die Integration von anderen, dem FIPA-Standard folgenden Systemen (für eine ausführliche Beschreibung potenzieller agentenbasierter Anwendungen im Gesundheitswesen siehe Kirn et al. 2006b, 199-337) erlaubt. Ausbaupotenziale der JADE-Plattform bzw. ihrer Erweiterungen JADE-LEAP bzw. JADE-S wurden bereits in den Abschnitten 7.6, 7.8 sowie 7.10 beschrieben.

Neben den in der vorliegenden Arbeit identifizierten Vorteilen von agentenbasierten Systemen lassen sich bei solchen Anwendungen auch systeminhärente Nachteile identifizieren: Mit

einem verteilten System, wie es Agentensysteme darstellen, werden entsprechend der Anzahl der Agenten potenzielle Sicherheitsrisiken eingeführt. Um erforderliche Sicherheitseigenschaften zu berücksichtigen, wurde die Erweiterung JADE-S entwickelt und in der umgesetzten Implementierung eingesetzt. Da mit dieser Erweiterung die möglichen Sicherheitseigenschaften nicht in allen Szenarien technisch unterstützt werden, würde eine entsprechende Erweiterung von JADE-S Abhilfe schaffen. Obschon mit JADE-S insgesamt wesentliche Sicherheitseigenschaften umgesetzt werden können, bleibt dennoch die Gefahr der Manipulation von Hosts durch mobile Agenten und umgekehrt bestehen. Eine Lösung dieses Problems wurde bisher nur durch Verzicht auf mobile Agenten erreicht (Kamel Boulos et al. 2006). Da gerade mobile Agenten wesentliche Vorteile besitzen (Lange/Oshima 1999) und auch die Mobilität im Gesundheitswesen inhärent gegeben ist (Niemann/Eymann 2006), können Bestrebungen zur Erhöhung der Sicherheit von insbesondere mobilen Agenten[105] identifiziert werden.

Das in einem Anwendungsfall (Schweiger et al. 2006b) beschriebene Teilsystem der mobilen Portierung zeigt die grundsätzliche Eignung für die Verbesserung der Informationslogistik in Krankenhäusern, es bleiben aber insbesondere noch Erfahrungen im Feldbetrieb zu sammeln. Einem dafür geeigneten Einsatz des Prototyps in der klinischen Erprobung steht jedoch die Tatsache entgegen, dass die im Referenzklinikum eingesetzten Informationssysteme keine durchgehende Informationssicht anbieten, sondern im Wesentlichen nur die Dokumentensicht unterstützen. Aus diesem Grund sind Erprobungssysteme mit manuell kodierten vollständigen Informationen zu konstruieren. Für deren Integration in das Agentensystem eignet sich der in der vorliegenden Arbeit beschriebene Ansatz; denn die Schnittstellen zwischen dem AMD und den externen Systemen sind skalierbar ausgelegt.

Zusammenfassend kann festgestellt werden, dass sich die gewählte Software-Architektur zur Lösung von informationslogistischen Defiziten im Gesundheitswesen grundsätzlich einsetzen lässt. Bei der prototypischen Implementierung konnte eine Reihe von technischen Defiziten identifiziert werden, die in weiteren Entwicklungsstufen der Plattformen und Bibliotheken zu beseitigen sind.

Außerdem lässt sich festhalten, dass eine Informationsaggregation nur dann erfolgreich implementiert werden kann, wenn die zusammenzuführenden Dokumente in einem geeigneten Format vorliegen, das eine automatisierte Weiterverarbeitung zulässt. Dies ist in der aktuellen Situation im Gesundheitswesen jedoch meist noch nicht vollständig gegeben.

7.13 Ergebnisse

Aus den bisherigen Beschreibungen werden in den folgenden Abschnitten 7.13.1 und 7.13.2 potenzielle Entwicklungsperspektiven für zukünftige Krankenhausinformationssysteme sowie der durch den Einsatz von AMDen induzierte Mehrwert abgeleitet.

7.13.1 Entwicklungsperspektiven für ein Krankenhausinformationssystem

Der Generalisierungsanspruch der Ergebnisse der vorliegenden Arbeit kann untermauert werden, indem aus dem Vergleich mit einem ausgewählten, gegenwärtig eingesetzten KIS Ent-

[105] Für weitere Möglichkeiten zur Erhöhung der Sicherheit im Kontext von mobilen Agenten wird auf die Arbeiten von *Roth/Jalali* (2001), *Binder/Roth* (2002), *Pinsdorf et al.* (2002), *Roth* (2002) und *Peters* (2006) sowie die folgenden Seiten verwiesen: http://www.semoa.org, zugegriffen am 16.02.2007.

wicklungsperspektiven für zukünftige Informationssysteme im Gesundheitswesen abgeleitet werden. Damit fließen die in der vorliegenden Arbeit vorgestellten Konzepte in die Wissensbasis (siehe dazu Abbildung 1.5-1) ein.

Für den Vergleich wird in der vorliegenden Arbeit das Informationssystem SAP IS-H bzw. i.s.h.med ausgewählt. Die Gründe hierfür liegen in der weiten Verbreitung der genannten Informationssysteme (Krcmar et al. 2007, 24-25) und in dem Einsatz dieses KIS in dem analysierten Referenzklinikum (siehe dazu Abschnitt 6.6.4.3). Dieses KIS wird, aus der Perspektive der in der vorliegenden Arbeit abgeleiteten Erkenntnisse anhand der Kriterien Plattform, Architektur, Funktionalität und Interoperabilität hinsichtlich seiner Entwicklungspotenziale beleuchtet:

- *Plattform*: Das betrachtete KIS ist auf eine Bedienung mit Desktop-Systemen ausgelegt. Eine zukünftige Portierung auf mobile Endgeräte unter Berücksichtigung der beschränkten Ressourcen und der gegebenen Möglichkeiten der Benutzerinteraktionen kann wesentliche Vorteile im operativen Betrieb einräumen. Für weitere Ausführungen zur Vorteilhaftigkeit des Einsatzes von mobilen Endgeräten siehe Abschnitt 4.6.8. Für eine Portierung auf unterschiedliche Endgeräte würde insbesondere auch eine Unterstützung von heterogenen Plattformen vorteilhaft sein, wie sie z.B. in der Java-Laufzeitumgebung gegeben ist.
- *Architektur*: Das im Anwendungsfall betrachtete KIS kann auf IS-Architekturebene über die Funktionalität von SAP Netweaver™ als offene Architektur betrachtet werden. Die prototypische Implementierung zeigt weiterhin, dass mit diesem KIS eine Informationsintegration ermöglicht wird, womit eine Einordnung der Gesamtapplikation als verteilte IS-Architektur vorgenommen werden kann. Trotz der damit auf der IS-Architekturebene gegebenen Flexibilität lassen sich beim betrachteten KIS auf Software-Architekturebene Defizite identifizieren, weil es als Client-Server-Modell implementiert ist: Dementsprechend lassen sich die Nachteile dieser Architektur (siehe dazu Abschnitt 3.4.2.1) auf das betrachtete KIS übertragen. Insbesondere kann dabei die nur unzureichend gegebene Flexibilität, induziert durch geringe Modularisierung und wenig ausgeprägte lose Kopplung, identifiziert werden. Die Flexibilität ist jedoch in Integrationsszenarien, wie sie bei der integrierten Versorgung vorliegen, ein wesentliches Kriterium. Für zukünftige Informationssysteme ist damit die Flexibilität auf den Ebenen der IS- und Software-Architektur durch die Wahl geeigneter Rahmenbedingungen zu berücksichtigen.
- *Funktionalität*: Hinsichtlich Funktionalität verhält sich das gewählte KIS lediglich reaktiv (zur Reaktivität von bisherigen Informationssystemen im Gesundheitswesen siehe Zachewitz 2004b). Die Anforderungen im Gesundheitswesen fordern jedoch als Ergänzung dazu proaktives und flexibles Verhalten. Darüber hinaus ist wegen der Ausrichtung von Prozessen an Patienten, auch über Institutionsgrenzen hinaus, eine geeignete Funktionalität erforderlich, die das Informationssystem auf der Basis der Eigenschaften des gewählten Behandlungsprozesses steuert. Zur Realisierung dieser Anforderungen sind geeignete Mechanismen auf Software-Ebene erforderlich. Weiterhin induziert eine Ausrichtung von Prozessen am Patienten auch dessen Integration in den Behandlungsprozess.
- *Interoperabilität*: Zur Interoperabilität sind geeignete medizinische Standards jeweils in ausreichender Breite zu unterstützen. Dabei sollte der Fokus auf etablierten sowie offe-

nen und weniger auf proprietären Standards liegen. Dazu sind Standards auf syntaktischer und semantischer Ebene erforderlich. Es ist jedoch nicht ausschlaggebend, dass eine Vielzahl unterschiedlicher Standards unterstützt wird. Vielmehr ist eine Ausrichtung an etablierten Standards aus den jeweils genannten Ebenen zu erreichen. Die Integration der unterschiedlichen Standards wird als Aufgabe der Virtualisierungsschicht oberhalb der Ebene der Informationssysteme im Gesundheitswesen betrachtet und muss deshalb vom eigentlichen KIS nicht implementiert werden. Zur Interoperabilität von heterogenen Informationssystemen im Rahmen einer virtuellen Patientenakte ist weiterhin eine geeignete Kommunikationsinfrastruktur vorauszusetzen, die den Zugriff auf Daten aus den verteilten Informationssystemen erlaubt, falls die geforderte Authorisierung in den Informationssystemen gegeben ist. Diese Infrastruktur wird durch die TI für die eGK gegeben sein.

Mit den beschriebenen Aspekten zu Weiterentwicklungsperspektiven für Informationssysteme im Gesundheitswesen wurde zusammenfassend gezeigt, dass die vorliegende Arbeit wissenschaftlichen Ansprüchen genügt: Über eine Abstraktion werden allgemeine Weiterentwicklungsmöglichkeiten abgeleitet. Damit wird der Generalisierungsanspruch an die Ergebnisse der vorliegenden Arbeit (siehe dazu auch Tabelle 1.9-2) erfüllt. Diese Ausführungen fungieren als Ausgangspunkt für die Ableitung des Mehrwertes der in der vorliegenden Arbeit beschriebenen Architektur für aktive medizinische Dokumente, welcher im folgenden Abschnitt 7.13.2 beschrieben wird.

7.13.2 Mehrwert durch aktive medizinische Dokumente

Entsprechend der Klassifizierung nach den Kategorien Plattform, Architektur, Funktionalität und Interoperabilität aus Abschnitt 7.13.1 wird im Folgenden der Mehrwert beschrieben, der durch die Einführung von aktiven medizinischen Dokumenten induziert werden kann, wie er in dem beschriebenen Anwendungsfall bereits untermauert wurde:

- *Plattform*: Das Konzept der AMDe ist auf einer Java basierten Agentenplattform realisiert. Damit ist die Grundlage für eine Plattformunabhängigkeit der Anwendung implizit gegeben. Außerdem ist mit der Java-Laufzeitumgebung eine solche Plattform verfügbar, die auf die Leistungsfähigkeit und Eigenschaften der Endgeräte individuell angepasst werden kann. Insbesondere kann ein breites Spektrum von mobilen Endgeräten unterstützt werden. Detaillierte Ausführungen dazu sind in Abschnitt 6.3.1 beschrieben.
- *Architektur*: Das AMD ist als zusätzliche Virtualisierungsschicht oberhalb der existierenden Informationssystemlandschaft eingeführt. Damit können in einem AMD als virtuelle Patientenakte relevante Daten aggregiert werden, womit die IS-Architektur verteilt gestaltet ist. Die in der prototypischen Implementierung eingesetzte Agentenplattform ist FIPA kompatibel und erlaubt so die Integration anderer Plattformen, die diesem Standard folgen. Weiterhin wird über die Open Source-Eigenschaft der gewählten Plattform JADE auch die Ankopplung anderer Technologien ermöglicht, z.B. die Integration von Web Services über WSIG. Damit kann eine IS-Architektur auf der Basis des AMD-Konzepts als offen betrachtet werden. Insgesamt kann so die Flexibilität auf IS-Architekturebene erreicht werden. Die in der vorliegenden Arbeit beschriebene Auswahl für die Ankopplung unterschiedlicher Applikationen demonstriert, dass in einem AMD heterogene Anwendungen integriert werden können. Somit wird eine geeignete Basis für weitere Applikationen zur Verfügung gestellt (siehe dazu Abschnitt 8.3). Die gewählte agentenbasierte Architektur erlaubt mit einer geeigneten Erweiterung (siehe dazu Abschnitt 3.3) die Realisierung einer Peer-to-Peer-Architektur (siehe Abschnitt 3.2.3.4). Damit können die Nachteile von Client-Server-Anwendungen beseitigt und da-

mit die ambiente Intelligenz unterstützt werden. In der medizinischen Domäne erweist sich dieses Konzept als besonders vorteilhaft: Den unterschiedlichen Rollenträgern können geeignete persönliche Assistenten (für grundsätzliche Darstellungen siehe Wilczek 2007, 229-232) zur Verfügung gestellt werden. Die Basis für diese Assistenten können aktive medizinische Dokumente liefern. In diesen sind, analog z.B. zur Datenkapselung in der Objektorientierung (siehe dazu Abschnitt 2.2.5), relevante Informationen gekapselt. Damit wird das Prinzip der Datenkapselung auf der Ebene von Programmkonstrukten in den Bereich der Applikationslogik übertragen (für eine Beschreibung dieser Idee im Zusammenhang mit einer SOA siehe Richter/Haller/Schrey 2005, 413). Dementsprechend können über ein AMD Dienste im Sinne einer SOA zur Verfügung gestellt werden. Auch zur Realisierung von emergenten Systemen, die flexibel auf Änderungen in der Umwelt reagieren, können Software-Agenten einen Beitrag zur Realisierung liefern. Die dadurch ermöglichte Selbstorganisation entsprechend dem Prinzip des Organic Computing (siehe Abschnitt 3.3.4) erlaubt die Beherrschung von zunehmend komplexer werdenden Software-Systemen. Die Virtualisierung von Inhalten erfordert per se keine persistente Datenhaltung. Insofern werden Daten in der AMD-Architektur nicht zentral z.B. in einer Datenbasis der TI vorgehalten, sondern werden nach Bedarf transient zur Verfügung gestellt. Mit diesem Ansatz können somit informationslogistische Defizite beseitigt werden, ohne jedoch Daten ggf. entgegen dem Wunsch der Patienten zentral ablegen zu müssen. Damit besteht auch kein zusätzlicher Aufwand für die Bereitstellung und den operativen Betrieb einer zentralen Datenbasis innerhalb der TI für die eGK. Um die Performanz der Datenaggregation zu verbessern sowie die Daten auch bei fehlender Konnektivität einsehen zu können, können jedoch Daten aus dem AMD auch persistent in den lokalen Informationssystemen abgelegt bzw. Patienten z.B. in der Form von tragbaren Speichergeräten für die eigene Einsichtnahme zur Verfügung gestellt werden. Zusammenfassend kann festgestellt werden, dass sich agentenbasierte Systeme auf der Ebene der IS-Architektur wegen der durch den Einsatz von Standards induzierten Offenheit und der per se gegebenen Verteiltheit und auf der Software-Architekturebene wegen ihrer ausgeprägten Modularisierung und losen Kopplung besonders hinsichtlich ihrer Flexibilität auszeichnen. Die lose Kopplung wird im Vergleich zu anderen Architekturansätzen insbesondere durch die nachrichtenbasierte Kommunikation auf der Grundlage einer Ontologie erreicht. Damit sind solche Systeme prädestiniert für Integrationsszenarien, wie sie in der integrierten Versorgung angestrebt werden.

- *Funktionalität*: Die in der vorliegenden Arbeit gewählte Architektur erlaubt die Ankopplung eines Portals für Patienten, in das letztere selbstständig Daten eingeben und über das Daten aus dem AMD aggregiert werden können. Damit wird die bei *Waegemann* (1999) beschriebene Patientenorientierung erreicht. Dies ermöglicht schließlich die aktive Integration des Patienten in seinen Behandlungsprozess, wodurch eine höhere Compliance (Giglio et al. 1978; Essex/Doig/Renshaw 1990) und letztendlich eine verbesserte Behandlungsqualität (Ball/Lillis 2001; Cimino/Patel/Kushniruk 2001; Schwarze et al. 2005b, 193) erwartet werden können. Durch die Integration der Prozesssteuerung in das AMD ist die Steuerung der Anwendung entlang von Prozessen gegeben. Die Definitionen solcher Prozesse können auch die Grenzen von Institutionen überschreiten. Damit richten sich insgesamt die Funktionalitäten des Informationssystems und seine Bedienung an Prozessen aus. Darüber kann auch eine proaktive Funktionalität unterstützt werden, indem Informationen entsprechend dem informationslogistischen Prinzip vor dem Informationsbedarf zur Verfügung gestellt werden. Weiterhin kann eine flexible Anpassung des Informationssystems an veränderte Prozes-

se vorteilhaft durch die Rekomposition von Prozessschritten vorgenommen werden. Insgesamt kann über die Prozessorientierung eine vorteilhafte Konfiguration des Informationssystems und seiner Funktionalitäten vorgenommen werden. Auf der Basis von flexiblen Architekturen für Agentensysteme können auch flexible Dienstleistungen von Agenten angeboten werden. Die flexible Reaktion auf Veränderungen in der Umwelt, in der vorliegenden Arbeit durch einen geänderten Gesundheitszustand gegeben, kann vorteilhaft mit Agentensystemen umgesetzt werden. Wird eine geeignete flexible Architektur in einem Informationssystem gewählt, können z.B. über die Konzepte des Organic Computing oder der ambienten Intelligenz auch proaktive Dienste zur Verfügung gestellt werden. Wie die Ausführungen zur Architektur in diesem Abschnitt zeigen, können agentenbasierte Systeme dafür vorteilhaft eingesetzt werden.

- *Interoperabilität*: Auf der Basis der TI für die eGK können mit einem AMD geeignete Mehrwertdienste angeboten werden. Damit können letztendlich die im Gesundheitswesen gewünschten Verbesserungen hinsichtlich Effektivität und Effizienz erreicht werden. Ausgewählte Funktionalitäten zur Interoperabilität auf syntaktischer und semantischer Ebene wurden in der vorliegenden Arbeit umgesetzt. Damit kann mit einem AMD auf der Ebene der Interoperabilität ein wesentlicher Beitrag zur Umsetzung der Vision der durchgängigen, integrierten Versorgung (siehe dazu Abschnitt 4.2.7) geleistet werden.

Die Tragfähigkeit der prototypischen Implementierung konnte in der vorliegenden Arbeit auf unterschiedlichen Ebenen demonstriert werden. Aus den dabei gewonnenen Erkenntnissen können Entwicklungspotenziale für Informationssysteme im Gesundheitswesen und ein durch das AMD-Konzept induzierter Mehrwert abgeleitet werden. Daraus folgt die Beantwortung der in Abschnitt 1.4 beschriebenen Forschungsfrage 4.

7.14 Zusammenfassung

In Abschnitt 7 wurde eine Bewertung der Ergebnisse der vorliegenden Arbeit vorgenommen. Dabei konnte insbesondere festgestellt werden, dass die Anforderungen an Informationssysteme im Gesundheitswesen durch die beschriebenen Implementierungen prinzipiell erfüllt werden. Dennoch wurde auch eine Reihe von Mängeln identifiziert, die in weiteren Arbeiten beseitigt werden können.

Weiterhin wurden insbesondere die Entwicklungsperspektiven für zukünftige Informationssysteme im Gesundheitswesen abgeleitet. Bei der Betrachtung des daraus resultierenden Mehrwertes von aktiven medizinischen Dokumenten zeigt sich, dass letztere diejenigen Eigenschaften besitzen, die für die Unterstützung der identifizierten Entwicklungsperspektiven geeignet sind.

8 Kritische Würdigung und Ausblick

8.1 Überblick

In Abschnitt 8 werden die wesentlichen Ergebnisse der vorliegenden Arbeit zusammengefasst und es wird ein Ausblick für weitere Arbeiten gegeben: Die Forschungsfragen aus Abschnitt 1.4 werden in Abschnitt 8.2 erneut aufgegriffen und beantwortet, indem die Resultate der vorliegenden Arbeit dargestellt werden. In Abschnitt 8.3 wird der Erkenntnisfortschritt der Arbeit beschrieben und bewertet. Abgeschlossen wird dieser Abschnitt mit einer Darstellung von weiteren, möglichen Arbeiten in Abschnitt 8.4.

8.2 Zusammenfassung

Der Aufbau der vorliegenden Arbeit orientiert sich an den in Abschnitt 1.4 beschriebenen forschungsleitenden Fragestellungen. Letztere werden zusammen mit den erzielten Ergebnissen im Folgenden beschrieben:

Ausgangspunkt für die vorliegende Arbeit ist eine umfangreiche Literaturanalyse zur Beschreibung des Status quo im Gesundheitswesen sowie zur Identifikation von informationslogistischen Defiziten. Damit soll die folgende Forschungsfrage beantwortet werden:

1. Was sind der Status quo und die Anforderungen im Gesundheitswesen?

Besonders prominent zeigten sich bei der Literaturanalyse die Eigenschaften der Komplexität, Verteiltheit, Patientenorientierung und Dynamik, die die Konstruktion von geeigneten Informationssystemen erschweren. Die Ergebnisse der Analyse wurden in dem in Tabelle 8.2-1 dargestellten Anforderungskatalog für ein Informationssystem komprimiert. Diese Anforderungen sind von einem Informationssystem im Gesundheitswesen zu erfüllen.

Anforderung	Ebene
Anforderung 1: Integration in eine geeignete Kommunikationsplattform	Plattform
Anforderung 2: Unterstützung von syntaktischen Standards	Syntax
Anforderung 3: Bereitstellung einer eindeutigen Patientenidentifikation	Kontext
Anforderung 4: Unterstützung von semantischen Standards	Semantik
Anforderung 5: Anwendungssäulen Informationslogistik, Entscheidungs- und Prozessunterstützung	Anwendung
Anforderung 6: Domänenbereiche Informationsmedien, Flexibilität, Unterstützung der Kommunikation, Systemnutzung und Patientenorientierung	Charakteristika der Domäne

Tabelle 8.2-1: ***Anforderungskatalog für ein Informationssystem im Gesundheitswesen***
Quelle: Eigene Darstellung

Um aus den genannten Anforderungen eine geeignete Implementierungstechnologie bzw. Architektur für eine Realisierung von Informationssystemen abzuleiten, wurden zunächst grundlegende Programmiertechniken der Informatik mit zunehmender Abstraktionsstufe beschrieben. Insbesondere wurden die Unterschiede zwischen der Agententechnologie und der Objektorientierung als gegenwärtigem Status quo in der Software-Entwicklung herausgearbeitet. Es konnte insgesamt demonstriert werden, dass mit Software-Agenten unter gewissen Rahmenbedingungen die stetig zunehmende Komplexität von Software-Systemen vorteilhaft beherrscht werden kann.

Für umfangreiche Systeme werden etablierte Architekturmuster eingesetzt. Deshalb wurden in der vorliegenden Arbeit gängige Architekturen, insbesondere für verteilte Systeme und zur Informationsintegration, beleuchtet. Dabei konnte intersubjektiv nachvollziehbar dargestellt werden, dass sich agentenbasierte Systeme hinsichtlich Flexibilität zur Laufzeit gegenüber anderen Ansätzen besonders auszeichnen. Diese Flexibilität ist induziert durch die ausgeprägte lose Kopplung und die besonderen Möglichkeiten zur Modularisierung. Da erstere für Integrationsszenarien, wie sie für die integrierte Versorgung erforderlich sind, wesentlich ist, eignen sich agentenbasierte Systeme deshalb als Grundlage für die Konstruktion geeigneter Informationssysteme.

Diese Argumentation wird dadurch untermauert, dass sich das Gesundheitswesen geeignet als CAS modellieren lässt. Weiterhin konnte begründet werden, dass Software-Agenten die Voraussetzungen besitzen, ein CAS adäquat durch Konstrukte der Software-Technik abzubilden. Dieser Zusammenhang erlaubt den transitiven Schluss, dass sich auch das Gesundheitswesen geeignet durch Informationssysteme auf der Basis von Software-Agenten darstellen lässt.

Diese Argumentationskette bildet die Grundlage für die Beantwortung der folgenden Forschungsfrage:

2. Welches Lösungskonzept überwindet die identifizierten Defizite?

Um die prototypische Implementierung in einen Anwendungsfall einzubetten, wird ein Behandlungsprozess im Referenzklinikum ausgewählt, der einerseits aus medizinischer Hinsicht relevant ist. Bei der Modellierung eines Prozessausschnittes zeigt sich, dass dieser andererseits einer angemessenen Unterstützung durch ein Informationssystem bedarf.

Im Zentrum des in der vorliegenden Arbeit beschriebenen Lösungskonzepts stehen aktive medizinische Dokumente, welche als komposite Software-Agenten implementiert sind und relevante Informationen über den Patienten kapseln. Damit wird eine virtuelle Patientenakte implementiert, die auf der Ebene oberhalb der bisherigen Informationssystemlandschaft eingesetzt werden kann. Das AMD erlaubt weiterhin die Integration von weiteren Anwendungen über die Einbettung von internen Agenten.

Die Konstruktion eines Prototyps wird durch die folgenden Aspekte geleitet: Die Domäneneigenschaften im gewählten Behandlungsprozess werden anhand der enthnografischen und partizipierenden Methode Needs Driven Approach analysiert. Die Ergebnisse fließen in die

weitere Konstruktion des Agentensystems nach der Methode ArBaCon ein. Die Implementierung wird dabei durch die Eigenschaften des Prototypenbaus fokussiert.

Am Beispiel der Informationsaggregation wird der gesamte ArBaCon-Entwicklungsprozess mit seinen jeweiligen Schritten und Ergebnissen erläutert. Das Design zur Realisierung der weiteren Funktionalitäten zur Einbettung in die TI, zum Scheduling, zur Prozessunterstützung sowie für die Portierung der virtuellen Patientenakte auf ein mobiles Endgerät und die Konstruktion eines Patientenportals wird nur anhand der jeweiligen Ergebnisse beschrieben.

Die Ergebnisse dieser Arbeitsschritte und damit die Beantwortung der Forschungsfrage 2 sind in Tabelle 8.2-2 zusammengefasst.

Abstrakter Anforderungskatalog aus Abschnitt 4.12	**Eigenschaften des Lösungskonzepts (Anforderungen aus Abschnitt 6.3.6)**
Anforderung 1	• Implementierung auf der Basis von Java (Anforderung 12) und der Agentenplattform JADE (Anforderung 19) • Simulation relevanter Komponenten der TI (Anforderungen 1, 2 und 3) • Portierung der virtuellen Patientenakte auf ein mobiles Endgerät (Anforderungen 13, 14 und 19) unter Berücksichtigung der besonderen Eigenschaften dieser Endgeräte
Anforderung 2	• Berücksichtigung etablierter syntaktischer Standards • Datenversand im Format HL7 CDA (Anforderung 4)
Anforderung 3	AMD-Konzept einschließlich Kapselung der Patienteninformationen (Anforderung 5) entsprechend der EAI-Datenintegration (Anforderung 15)
Anforderung 4	• Abbildung von Nachrichteninhalten auf Ontologiekonzepte (Anforderung 7) • Berücksichtigung etablierter semantischer Standards • Integration von LOINC-Codes in die HL7 CDA-Dokumente (Anforderung 6)
Anforderung 5	• Informationsaggregation im AMD (Anforderung 8) • Prozessintegration im AMD im Sinne der EAI (Anforderung 15) • Unterstützung eines ausgewählten klinischen Pfades (Anforderung 9) zur Unterstützung der Prozessorientierung (siehe Abschnitt 4.2.4) und einer potenziellen Analyse und Weiterentwicklung von Behandlungspfaden (Berger 2004, 50-53) • Modularisierte Umsetzung des AMD zur Integration unterschiedlicher Funktionalitäten (Anforderung 17) • Berücksichtigung von etablierten Architekturprinzipien (Anforderungen 15 mit 18)
Anforderung 6	• Verteilter Scheduling-Algorithmus mit flexibler Reaktionsfähigkeit (Anforderung 10) • Ankopplung eines Patientenportals im Sinne eines eKiosks (Anforderungen 11, 17 und 19) auf der Basis von Web Services • Verschlüsselung und Signatur (Anforderungen 19 und 20)

Tabelle 8.2-2: ***Zusammenfassung des agentenbasierten Lösungskonzepts für ein Informationssystem im Gesundheitswesen***
Quelle: Eigene Darstellung

Auf der Basis des dargestellten Lösungskonzepts werden Elemente auf Implementierungsebene beschrieben. Mit der detaillierten Beschreibung der Implementierung entlang ihren Entwurfsentscheidungen wird Forschungsfrage 3 beantwortet:

3. Welche Elemente und Eigenschaften besitzt eine Architektur für eine prototypische Implementierung, die die Tragfähigkeit des erarbeiteten Lösungskonzepts demonstriert?

Die Tragfähigkeit des erarbeiteten Lösungskonzepts wird durch eine agentenbasierte Implementierung mit umfangreicher Funktionalität demonstriert. Somit können die in Tabelle 8.2-3 genannten Architekturelemente hinsichtlich ihrer Eignung verifiziert werden.

Abstrakter Anforderungskatalog aus Abschnitt 4.12	**Architekturelemente zur Erfüllung der Anforderungen aus Abschnitt 6.3.6**
Anforderung 1	• Implementierung der Komponenten Konnektor (HConnectorAgent, GPConnectorAgent, Anforderung 1) und Kartenlesegerät (CardReaderGuiAgent, Anforderung 2) sowie geeigneter Kommunikationsprotokolle zur Kommunikation • Komponente zur Protokollierung (Klasse PProtocol, Methode protocolling) der letzten 50 Zugriffe auf die Gesundheitskarte (Anforderung 3) und ihre Speicherung im lokalen Dateisystem • Implementierung mit Java (Anforderung 12) und JADE (Anforderung 19) • Komponenten zur Implementierung der mobilen Anwendung zur Informationsaggregation (PersonalAssistantGuiAgent, PersonalAssistantGui, Anforderung 13) auf der Basis von JADE-LEAP (Anforderung 19), Konfiguration J2ME/CDC/Personal Profile, Gestaltung einer Benutzeroberfläche mit den gegebenen Funktionalitäten von Java AWT (Anforderung 14)
Anforderung 2	• Einbettung und Strukturierung der Patientendaten in Ontologiekonzepte (Konzept MedicalDocument, Anforderung 7) • Definition von adäquaten XSL-Transformationen zur Abbildung von HL7 CDA- auf HTML-Dokumente (Desktop-Anwendung) • Implementierung eines Parsers zur Abbildung von HL7 CDA-Dokumenten auf geeignete Elemente von Java AWT (mobile Anwendung)
Anforderung 3	• Datenaggregation aus verteilten Informationssystemen in dem AMD-Konstrukt aus elementaren Software-Agenten (DataRetrievalManager, DataWrapper, ViewManager, AppointmentMonitor, AppointmentManager, Subscriber, ProcessMasterAgent, Anforderung 5) unter Berücksichtigung einer globalen Patientenidentifikation • DataRetrievalManager als einziger Eintrittspunkt in das AMD (ActiveMedicalDocument, Anforderung 15)
Anforderung 4	• Einbettung und Strukturierung der Patientendaten in Ontologiekonzepte (Konzept MedicalDocument, Anforderung 7) • Integration von LOINC-Codes in die HL7 CDA-Dokumente (Anforderung 6)
Anforderung 5	• (Teilw. simulierte) Extraktion von Daten aus SAP IS-H bzw. i.s.h.med über JCo (Klasse SAPJCoConnection sowie ihre Methoden, Anforderung 8) und geeignete Wrapper-Agenten (Klasse HISWrapper sowie ihre Methoden, Anforderung 8) • Extraktion von Daten aus einem (simulierten) PVS über einen geeigneten Wrapper-Agent (GPISWrapper) • Implementierung einer agentenbasierten Prozessunterstützung mit den beiden Phasen der Prozessmodellierung und -ausführung (Anforderung 9) sowie einem persistenten Prozessinformationsspeicher, Transformation der AML-Beschreibung eines EPK-Modells in die ausführbare PML-Beschreibung (für eine Beschreibung der implementierenden Klassen siehe Abschnitt 6.8.7) • Zusammenführung von Teilprozessen zu einem Gesamtprozess, damit erfolgt die Prozessintegration im Sinne der EAI (Klasse ProcessImport und ihre Methoden, Anforderung 15) • Modellierung eines Beispielpfades (Anforderung 9) mit der Unterbreitung von Vorschlägen für das weitere Vorgehen im Prozess durch eine agentenbasierte Prozessunterstützung (für eine Beschreibung der implementierenden Klassen siehe Abschnitt 6.8.7) • Modularisierte Umsetzung des AMD über die Bereitstellung von Funktionalitäten durch dedizierte Agenten, Anbindung an ein Patientenportal über WSIG (Agent Subscriber, Anforderung 19) und das Simulationssystem SeSAm (Anforderung 17)

	• Berücksichtigung der Architekturprinzipien SOA/EAI (Anforderung 15), Schichtenarchitektur (Benutzerinteraktion (z.B. ViewManager, JPersonalAssistantGui), Anwendungslogik (z.B. AMDTaskAgent), Ankopplung von Ressourcen (HISWrapper, GPISWrapper), Anforderung 16), Modularisierung zur Integration des Simulationssystems SeSAm und eines Patientenportals auf der Basis von Liferay (Anforderung 17), flexible Komposition und Dekomposition (Starten der AMDTaskAgents durch den DataRetrievalManager, Anforderung 18) von Task-Agenten zur Aggregation der verteilten Informationen
Anforderung 6	• Implementierung eines verteilten Scheduling-Algorithmus (Anforderung 10) über die Modellierung von Ressourcen und Patienten sowie einer geeigneten Kostenfunktion (für eine Darstellung der erforderlichen Klassen siehe Abschnitt 6.8.5) • Vorschlag zur Realisierung eines eKiosks (Anforderung 11, für eine detaillierte Beschreibung der erforderlichen Klassen siehe Abschnitt 6.8.6) über die Ankopplung des AMD über Web Services bzw. die JADE-Erweiterung WSIG (Anforderungen 17 und 19) • (Eingeschränkte) Verschlüsselung und Signatur der Kommunikation zwischen den Konnektoren (HConnectorAgent, GPConnectorAgent) und Kartenlesegerät (CardReaderGuiAgent, Anforderungen 19 und 20)

Tabelle 8.2-3: ***Zusammenfassung des agentenbasierten Ansatzes für ein Informationssystem im Gesundheitswesen***
Quelle: Eigene Darstellung

Abschließend sind die Erkenntnisse der vorliegenden Arbeit in der Beantwortung der folgenden Forschungsfrage 4 zu explizieren:

4. *Welchen Mehrwert und Nutzen schafft das Konzept des aktiven medizinischen Dokuments?*

In der Evaluierung der vorliegenden Arbeit wurde dazu überprüft, ob die Anforderungen an die implementierte Anwendung erfüllt wurden: Dafür wurde ein Evaluierungsrahmenwerk eingeführt, auf dessen Grundlage die Bewertung der Ergebnisse der Arbeit erfolgte. Dabei zeigte sich, dass die Anforderungen aus dem erarbeiteten Anforderungskatalog erfüllt werden konnten. Weiterhin ließ sich feststellen, dass sich die besonderen Eigenschaften des Gesundheitswesens über Software-Agenten vorteilhaft auf die Ebene der Software-Technik projizieren lassen. Auch mögliche Auswirkungen eines ganzheitlichen Behandlungsprozesses auf Patienten und Leistungserbringer konnten aufgezeigt werden. Außerdem konnte über den Vergleich mit der gewählten Referenzimplementierung das Entwicklungspotenzial von Informationssystemen im Gesundheitswesen demonstriert werden. Dazu wurden die Kategorien Plattform, Architektur, Funktionalität und Interoperabilität identifiziert, die für zukünftige Informationssysteme grundlegende Bedeutung besitzen. Dabei sind jeweils die folgenden Eigenschaften von Bedeutung:

- *Plattform*: Unterstützung heterogener Plattformen und Endgeräte
- *Architektur*: Gewährleistung der Flexibilität auf den Ebenen der IS- und Software-Architekturen
- *Funktionalität*: Unterstützung der Proaktivität, Ausrichtung an Prozessen, Integration des Patienten
- *Interoperabilität*: Unterstützung etablierter syntaktischer und semantischer Standards sowie Integration in eine Kommunikationsinfrastruktur

Aufbauend auf diesen Ergebnissen wurde der Mehrwert durch aktive medizinische Dokumente zusammenfassend dargestellt. Dabei zeigte sich, dass das AMD-Konzept die Eigenschaften von zukünftigen Informationssystemen umsetzen kann:

- *Plattform*: Implementierung auf der Basis der Java-Laufzeitumgebung mit der Möglichkeit, unterschiedliche Konfigurationen für verschiedene Endgeräte zu erstellen
- *Architektur*: AMD als Virtualisierungsschicht und damit Realisierung einer verteilten IS-Architektur, Konstruktion eines offenen Systems induziert durch die Wahl einer FIPA kompatiblen Agentenplattform, agentenbasierte Umsetzung der ambienten Intelligenz und des Organic Computings, inhärente Flexibilität in einem MAS auf Software-Architekturebene, insgesamt hoher Grad an Flexibilität auf den Ebenen der IS- und Software-Architektur, keine zentrale Speicherung von Patientendaten
- *Funktionalität*: Integration des Patienten über eine Web basierte Schnittstelle, Steuerung von Informationssystemen entlang von Behandlungsprozessen, Rekonfigurierung von Informationssystemen durch eine geeignete Anpassung von Behandlungsprozessen und deren Abbildung auf eine ausführbare Beschreibung, flexible Reaktion auf Veränderungen in der Umwelt
- *Interoperabilität*: AMD als Mehrwertdienst der TI für die eGK, Berücksichtigung ausgewählter semantischer und syntaktischer Standards

Die technische Korrektheit der Implementierung konnte in geeigneten Labortests demonstriert werden: Das dabei verwendete Testverfahren lehnte sich an etablierte Vorgehensweisen des Software Engineering an.

Insgesamt kann festgestellt werden, dass die bei *Haux* (2005, 268) identifizierten Trends für die Entwicklung von Informationssystemen im Gesundheitswesen mit dem in der vorliegenden Arbeit beschriebenen Lösungskonzept realisiert werden können. Diese Trends und ihr Bezug zur vorliegenden Arbeit sind in Tabelle 8.2-4 zusammengefasst. Somit werden die Entwicklungsperspektiven bei *Haux* (2005, 268) durch die Ergebnisse der vorliegenden Arbeit bestätigt.

Trends	**Bezug zur vorliegenden Arbeit**
• Verschiebung von Papier basierter zu IT basierter Informationsverarbeitung • Zunahme von Daten	• Auf der Basis der digital vorliegenden Daten wird eine virtuelle Patientenakte zusammengestellt. • Software-Agenten besitzen das Potenzial der Informationsaggregation, -filterung und ihrer Personalisierung
Verschiebung von an Abteilungen orientierten Informationssystemen zu globalen Informationssystemen	Die virtuelle Patientenakte erlaubt die Datenaggregation aus verteilten Informationssystemen über die TI für die eGK.
Integration der Patienten in Informationssysteme	Entwurf eines Portalsystems für den Patienten in der Form eines eKiosks
Einsatz von Daten auch zur Steuerung des Gesundheitswesens und zur klinischen und epidemiologischen Forschung	Anonymisierte Weiterverarbeitung der aggregierten Daten der virtuellen Patientenakte z.B. durch das bei *Kamel Boulos et al.* (2006) beschriebene, agentenbasierte Verfahren

Fokussierung weniger auf technische als vielmehr Aspekte des Informationsmanagements	Potenzielle Berücksichtigung des Kontextes für den Informationsbedarf z.B. über die Ansätze von *Holzinger/Geierhofer/Errath* (2007) oder *Wilczek* (2007)
Integration unterschiedlicher Datentypen (alphanumerisch, Bild, molekular)	Möglichkeit zur Integration multimedialer Daten in der virtuellen Patientenakte
Unterstützung von Ubiquitous Computing	Portierung der virtuellen Patientenakte auf ein mobiles Endgerät

Tabelle 8.2-4: ***Entwicklungsperspektiven für Informationssysteme im Gesundheitswesen und ihr Bezug zur vorliegenden Arbeit***
Quelle: Eigene Darstellung, Zusammenfassung der Trends bei *Haux* (2005, 268)

Obwohl in der vorliegenden Arbeit einige technische, jedoch überwindbare Mängel, insbesondere bei der Umsetzung von Sicherheitseigenschaften, identifiziert werden, kann festgestellt werden, dass sich Software-Agenten für die Konstruktion von Informationssystemen im Gesundheitswesen eignen. Dies konnte mit ausgewählten, in der vorliegenden Arbeit beschriebenen Implementierungen untermauert werden. Insbesondere können mit der in der vorliegenden Arbeit beschriebenen Lösung die bei Haux (2005, 268) identifizierten Entwicklungsperspektiven von Informationssystemen im Gesundheitswesen geeignet unterstützt werden.

8.3 Beschreibung und Bewertung des Erkenntnisfortschrittes

Grundlage für die Beschreibung des Erkenntnisfortschrittes sind die Ausführungen zur Evaluierung in Abschnitt 7.3: Zusammen mit den Ergebnissen der durchgeführten Labortests in Abschnitt 7.4 konnte mit der prototypischen Implementierung in dieser Arbeit die Tragfähigkeit der AMD-Architektur als agentenbasierte Integrationskomponente untermauert werden. Dazu kann das AMD auf der Basis einer Virtualisierungsschicht oberhalb der Ebene der bisherigen Informationssystemlandschaft im Gesundheitswesen als Mehrwertdienst auf der Grundlage der TI für die eGK betrachtet werden. Damit gilt die TI, betrachtet als eine Service orientierte Architektur, als eine adäquate Basis zur Funktionalitätserweiterung durch Mehrwertdienste. Die angestrebte Informationsintegration konnte mit dem AMD-Konzept durch die in einem agentenbasierten System gegebene Flexibilität vorteilhaft umgesetzt werden. Somit können Software-Agenten zur Realisierung von geeigneten Informationssystemen im von Dynamik geprägten Gesundheitswesen eingesetzt werden. Die dafür erforderliche Flexibilität kann dabei auf den Ebenen der IS- und Software-Architekturen erzielt werden. Insbesondere können die identifizierten Entwicklungsperspektiven für ein Informationssystem im Gesundheitswesen vorteilhaft durch ein agentenbasiertes System erreicht werden.

Die in der vorliegenden Arbeit beschriebene Implementierung besitzt zwar nur den Charakter eines Prototyps. Für die Demonstration der Tragfähigkeit des beschriebenen Lösungsansatzes sowie die Identifikation von technischen Hürden und weiteren Arbeiten ist die Implementierung jedoch ausreichend.

8.4 Ausblick

In der vorliegenden Arbeit wurde mit dem AMD-Konzept eine Grundlage für weitere Arbeiten gelegt. Die beschriebenen Implementierungen zeigen bereits, dass ein AMD mit unterschiedlichen Anwendungen auf heterogenen Plattformen gekoppelt werden kann. Dennoch

lässt sich auch eine Reihe von Lücken identifizieren, die in weiteren Arbeiten zu schließen sind. Diese können anhand den jeweiligen Zielgruppen für die Anwendung differenziert werden: Gesundheitsdienstleister bzw. Akteure zur Steuerung des Gesundheitswesens (Roland Berger & Partner GmbH – International Management Consultants 1997, 11):

Mit dem AMD-Konzept kann u.a. eine virtuelle Patientenakte zusammengestellt werden. Im Design wurde zwar die Zusammenstellung eines unterschiedlichen Informationsbedarfs berücksichtigt, jedoch nicht in der Implementierung umgesetzt. Agenten können die Aufgabe der Aggregation von Informationen anhand von Profilen bzw. Ansichten (siehe dazu auch Abbildung 8.4-1, wobei Agenten als Sichtengeneratoren fungieren können) wahrnehmen. Ein Ansatz dazu ist bei *Wilczek* (2007, 229-232) beschrieben, der jedoch noch agentenbasiert anzupassen ist.

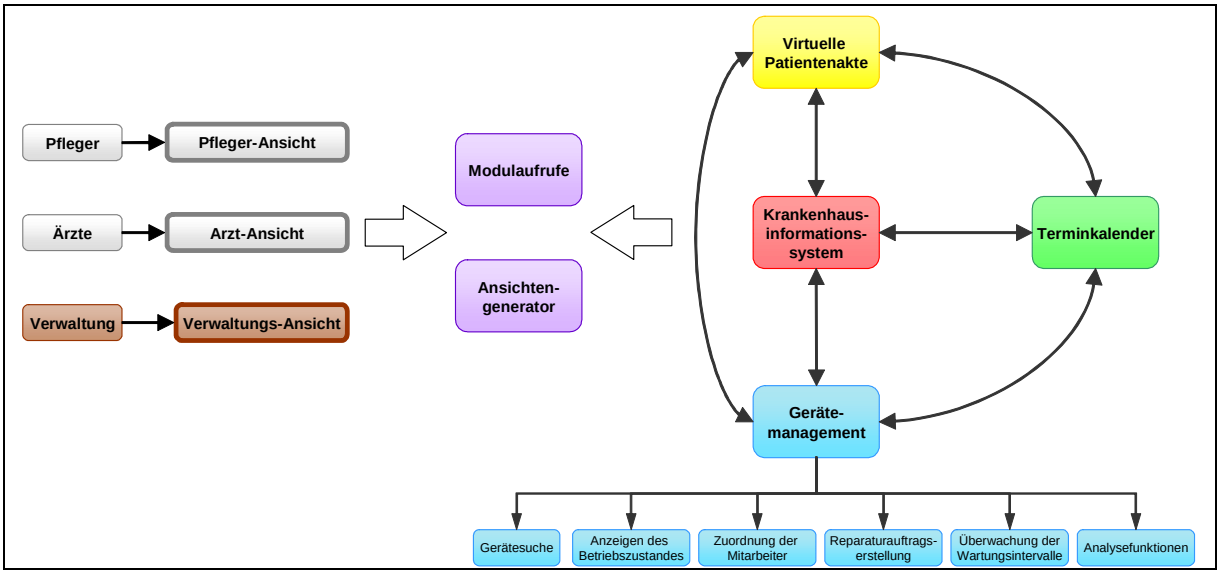

Abbildung 8.4-1: ***Gerätemanagement, KIS, virtuelle Patientenakte und Terminkalender***
Quelle: *Ortmann et al.* (2007, 40)

Werden die bei der Prozesssteuerung durch den jeweiligen Rollenträger gewählten Entscheidungen für die weitere Behandlung dokumentiert und automatisiert analysiert, kann diese Wissensbasis eine Grundlage für die evidenzbasierte Medizin bilden (siehe dazu auch Bergmann/Klinger/Winkler 2006, 89-90). Daraus können in einem weiteren Schritt Behandlungsempfehlungen für Krankheitsbilder vorgeschlagen werden. Erstere können damit entsprechend der sich stetig ändernden Wissensbasis der medizinischen Domäne angepasst werden.

Die nicht vollständige Implementierung von Sicherheitseigenschaften der prototypischen Applikationen wurde bereits in Abschnitt 7.7 diskutiert. Bedarf besteht insbesondere für die Bereitstellung geeigneter Erweiterungen, die die JADE-Funktionalität derart ergänzen, dass relevante Sicherheitseigenschaften wie die Plattform übergreifende Verschlüsselung erfüllt werden können.

Die beschriebene Integration von Daten aus dem gewählten KIS i.s.h.med kann durch zukünftige BAPI-Funktionalitäten erweitert werden. Dazu wurden in der Implementierung bereits entsprechende Erweiterungsmöglichkeiten eingeräumt. Wird die BAPI-Funktionalität in zukünftigen Versionen von i.s.h.med bzw. IS-H erweitert, sind in der vorliegenden Implementierung nur die entsprechenden Schnittstellen anzupassen.

Die in der vorliegenden Arbeit beschriebene Implementierung einer Scheduling-Lösung ist auf die Berücksichtigung von Terminen des medizinischen Personals sowie von Patienten beschränkt. Da die vorliegende Implementierung modular gestaltet ist, ist eine Erweiterung zur Integration von Lösungen zum Gerätemanagement (Leimeister/Schweiger/Krcmar 2006) grundsätzlich möglich. Da bei medizinischen Prozessen insbesondere auch relevante medizinische Geräte zur Verfügung stehen müssen, diese sich jedoch in unterschiedlichen Zuständen oder an unterschiedlichen Lokationen befinden können, ist eine automatisierte Lokalisierung, Abfrage des Betriebszustandes von verfügbaren Geräten sowie deren Integration in die Planung von Prozessen (siehe dazu auch Abbildung 8.4-1) sinnvoll.

Neben der bisher beschriebenen Zielgruppe der Leistungserbringer können auf der Basis des AMD-Konzeptes auch Applikationen zur Steuerung des Gesundheitswesens umgesetzt werden: Dafür wurden bei *Kamel Boulos et al.* (2006) bereits geeignete Ansätze erarbeitet. Mit einer adäquaten Kopplung des AMD-Konzepts mit den dort beschriebenen Ausführungen können auch diejenigen Akteure des Gesundheitswesens mit den relevanten Informationen versorgt werden, die für die Steuerung und für Entscheidungen im Gesundheitswesen (Roland Berger & Partner GmbH – International Management Consultants 1997, 11) verantwortlich sind. Dazu sind anonymisierte Datenanalysen auf den aggregierten Patientendaten durchzuführen.

Voraussetzung für die in diesem Abschnitt beschriebenen Szenarien ist die Vereinheitlichung von Standards im Gesundheitswesen. Dazu ist jedoch kritisch anzumerken, ob angesichts der existierenden Vielfalt eine Einigung auf gemeinsame Standards erzielt werden kann. Vielmehr kann davon ausgegangen werden, dass durch geeignete Transformationsabbildungen zwischen den Standards eine Konvergenz erreicht werden kann. Dabei können insbesondere Software-Agenten durch die formalen Möglichkeiten ihrer Kommunikationssprache einen Beitrag leisten, indem sie zwischen unterschiedlichen Standards vermitteln. In der Tat wird Software-Agenten im Allgemeinen aufgrund ihrer formalen, standardisierten und mächtigen Kommunikationssprache ACL das Potenzial zur Integration von heterogenen Software-Systemen attribuiert (Genesereth/Ketchpel 1994, 48-49). Die Darstellungen bei *Corera/Laresgoiti/Jennings* (1996), *Jennings et al.* (1996b) und *Perriollat et al.* (1996) demonstrieren in einer produktiv eingesetzten Implementierung – allerdings in einer anderen Domäne als dem Gesundheitswesen – dass Agentensysteme tatsächlich in der Lage sind, heterogene Informationssysteme zu integrieren. Dazu sind – angewendet auf das Gesundheitswesen – geeignete Transformationen zwischen den Standards zu bestimmen. Die bisher gegebenen Möglichkeiten der Integration von Standards sind in Abschnitt 4.9.4 beschrieben. Entsprechend den dortigen Ausführungen in den Abschnitten 4.9.4.3 mit 4.9.4.5 ist die Integration auf Konzept- und Terminologieebene bzw. die Zusammenführung von syntaktischer und semantischer Integration nicht vollständig automatisierbar bzw. wegen fehlender Konvergenz der Standards bis dato nicht gegeben.

In weiteren Arbeiten ist nach einer geeigneten Erweiterung der agentenbasierten Applikationen eine Evaluierung im operativen Betrieb erforderlich, um Aussagen über Akzeptanz, wahrgenommene Vor- und Nachteile, Usability, Wirtschaftlichkeit sowie notwendige soziotechnische Arrangements in der Aufbau- und Ablauforganisation zur Verbesserung der Gesamtperformanz im Anwendungsfeld treffen zu können.

Literaturverzeichnis

Adam, D. (1993): Produktions-Management. (7. Aufl.), Gabler, Wiesbaden 1993.

Adam, D. (1996): Planung und Entscheidung: Modelle-Ziele-Methoden. (4. Aufl.), Gabler, Wiesbaden 1996.

Adams, I.D.; Chan, M.; Cliord, P.C.; Cooke, W.M.; Dallos, V.; de Dombal, F.T.; Edwards, M.H.; Hancock, D.M.; Hewett, D.J.; McIntyre, N.; Somerville, P.G.; Spiegelhalter, D.J.; Wellwood, J.; Wilson, D.H. (1986): Computer Aided Diagnosis of Acute Abdominal Pain: A Multicentre Study. In: British Medical Journal, Vol. 293 (1986) Nr. 6550, S. 800-804.

Aktionsforum für Telematik im Gesundheitswesen der GVG (Gesellschaft für Versicherungswissenschaft und -gestaltung e.V., 2004): ATG-Managementpapier „Elektronische Patientenakte“. http://atg.gvg-koeln.de/xpage/objects/patientenakte/docs/4/files/MP_ePa_050118.pdf, zugegriffen am 13.07.2005.

Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, S. (1977): A Pattern Language: Towns/Buildings/Construction, Oxford University Press, New York 1977.

Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V. (2004): Web Services: Concepts, Architectures and Applications, Springer, Berlin, Heidelberg, New York 2004.

Anderson, J.G. (1997): Clearing the Way for Physicians' Use of Clinical Information Systems. In: Communications of the ACM, Vol. 40 (1997) Nr. 8, S. 83-90.

Anderson, R.J. (2001): Security Engineering: A Guide to Building Dependable Distributed Systems, John Wiley & Sons, Inc., New York u.a. 2001.

Arbeitsgemeinschaft SCIPHOX GbR mbH (2005): SCIPHOX-Webseite. http://www.sciphox.de/, zugegriffen am 13.12.2005.

Augustin, S. (1990): Information als Wettbewerbsfaktor: Informationslogistik – Herausforderung an das Management, Verlag TÜV Rheinland, Köln 1990.

avetana GmbH (2005): avetana-Webseite. www.avetana.de, zugegriffen am 07.03.2006.

Avison, D.; Young, T. (2007): Time to Rethink Health Care and ICT? In: Communications of the ACM, Vol. 50 (2007) Nr. 6, S. 69-74.

Awizen, M.; Paulussen, T.O. (2001): Modellierung von Kommunikationsprotokollen für die dezentrale, agentenunterstützte Koordination von Krankenhausprozessen. Konferenzbeitrag: Informatik 2001: Wirtschaft und Wissenschaft in der Network Economie – Visionen und Wirklichkeit, S. 883-888.

Baldi, B.; Hoyer, D.; Hofmann, G.R.; Brettreich-Teichmann, W.; Niemeier, J.; Gräslund, K.; Krcmar, H.; Schwabe, G.; Konrad, P.; Seibt, D. (1996): BTÖV – Eine Methode für die bedarfsgerechte Gestaltung von Telekooperation in der öffentlichen Verwaltung. In: EMISA Forum, Mitteilung der GI-Fachgruppe „Entwicklungsmethoden für Informationssysteme und deren Anwendung“, Fachgruppe EMISA (Hrsg.), (1996) Nr. 1, S. 31-35.

Baldry, M.; Cheal, C.; Fisher, B.; Gillett, M.; Huet, V. (1986): Giving Patients Their Own Records in General Practice: Experience of Patients and Staff. In: British Medical Journal, Vol. 292 (1986) Nr. 6520, S. 596-598.

Ball, M.J.; Lillis, J. (2001): E-health: Transforming the Physician/Patient Relationship. In: International Journal of Medical Informatics, Vol. 61 (2001) Nr. 1, S. 1-10.

Bartel, M.; Boyer, J.; Fox, B.; LaMacchia, B.; Simon, E. (2002): XML-Signature Syntax and Processing. http://www.w3.org/TR/xmldsig-core/, zugegriffen am 14.02.2007.

Bass, L.; Clements, P.; Kazman, R. (2003): Software Architecture in Practice. (2. Aufl.), Addison-Wesley, Boston u.a. 2003.

Bastian, T. (2005): Vertikale Prozess- und Systemanalyse im Gesundheitswesen und Konzeption sowie Implementierung eines Agenten basierten Prototyps. Diplomarbeit, Technische Universität München 2005.

Bates, D.W. (2000): Using Information Technology to Reduce Rates of Medication Errors in Hospitals. In: British Medical Journal, Vol. 320 (2000) Nr. 7237, S. 788-791.

Bates, D.W.; Gawande, A.A. (2003): Improving Safety with Information Technology. In: New England Journal of Medicine, Vol. 348 (2003) Nr. 25, S. 2526-2534.

Bates, D.W.; Kuperman, G.J.; Wang, S.; Gandhi, T.; Kittler, A.; Volk, L.; Spurr, C.; Khorasani, R.; Tanasijevic, M.; Middleton, B. (2003): Ten Commandments for Effective Clinical Decision Support: Making the Practice of Evidence-based Medicine a Reality. In: Journal of the American Medical Informatics Association, Vol. 10 (2003) Nr. 6, S. 523-530.

Bates, D.W.; Leape, L.L.; Cullen, D.J.; Laird, N.; Petersen, L.A.; Teich, J.M.; Burdick, E.; Hickey, M.; Kleefield, S.; Shea, B.; Vliet, M.V.; Seger, D.L. (1998): Effect of Computerized Physician Order

Entry and a Team Intervention on Prevention of Serious Medication Errors. In: Journal of the American Medical Association, Vol. 280 (1998) Nr. 15, S. 1311-1316.

Bates, D.W.; Miller, E.B.; Cullen, D.J.; Burdick, L.; Williams, L.; Laird, N.; Petersen, L.A.; Small, S.D.; Sweitzer, B.J.; Vliet, M.V.; Leape, L.L. (1999): Patient Risk Factors for Adverse Drug Events in Hospitalized Patients. In: Archives of Internal Medicine, Vol. 159 (1999), S. 2553-2560.

Bates, D.W.; O'Neil, A.C.; Boyle, D.; Teich, J.; Chertow, G.M.; Komaro, A.L.; Brennan, T.A. (1994): Potential Identifiability and Preventability of Adverse Events Using Information Systems. In: Journal of the American Medical Informatics Association, Vol. 1 (1994) Nr. 5, S. 404-411.

Bates, J. (1994): The Role of Emotion in Believable Agents. In: Communications of the ACM, Vol. 37 (1994) Nr. 7, S. 122-125.

Bates, J.; Loyall, B.A.; Reilly, S.W. (1992): An Architecture for Action, Emotion, and Social Behaviour. Konferenzbeitrag: 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1992), S. Martino al Cimino, Italien, S. 55-68.

Baud, R.H.; Lovis, C.; Rassinoux, A.-M.; Scherrer, J.-R. (1998): Alternative Ways for Knowledge Collection, Indexing and Robust Language Retrieval. In: Methods of Information in Medicine, Vol. 37 (1998) Nr. 4-5, S. 315-326.

Bauer, B.; Müller, J.P.; Odell, J. (2001): Agent UML: A Formalism for Specifying Multiagent Software Systems. In: International Journal on Software Engineering and Knowledge Engineering, Vol. 11 (2001) Nr. 3, S. 1-24.

Bauer, B.; Odell, J. (2005): UML 2.0 and Agents: How to Build Agent-Based Systems with the New UML Standard. In: Engineering Applications of Artificial Intelligence, Vol. 18 (2005) Nr. 2, S. 141-157.

Baujard, O.; Baujard, V.; Aurel, S.; Boyer, C.; Appel, R.D. (1998): MARVIN – A Multi-Agent Softbot to Retrieve Multilingual Medical Information on the Internet. In: International Journal of Medical Informatics, Vol. 23 (1998) Nr. 3, S. 187-191.

Beard, J.W.; Keck, B.; Peterson, T.O. (2005): Information Systems and Health Care VII: When Success Results in Failure: The Challenge of Extending the IT Infrastructure to Support Organ Procurement and Transplantation. In: Communications of the Association for Information Systems, Vol. 16, Article 26 (2005).

Becker, M.; Czap, H. (2006): Artificial Software Agents as Representatives of Their Human Principals in Operating-Room-Team-Forming. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 221-237.

Becker, M.; Heine, C.; Herrler, R.; Krempels, K.-H. (2003): OntHoS – An Ontology for Hospital Scenarios. In: Applications of Software Agent Technology in the Health Care Domain. Hrsg.: Moreno, A.; Nealon, J. Birkhäuser, Basel 2003, S. 87-103.

Behrens, G. (1993): Wissenschaftstheorie und Betriebswirtschaftslehre. In: Handwörterbuch der Betriebswirtschaft: Teilband 3. Hrsg.: Wittmann, W.; Kern, W.; Köhler, R.; Küpper, H.-U.; von Wysocki, K., 5 (Aufl.). Schäffer-Poeschel, Stuttgart 1993, S. 4763-4772.

Bellavista, P.; Stefanelli, C. (2000): CORBA Solutions for Interoperability in Mobile Agent Environments. Konferenzbeitrag: Distributed Objects and Applications, Antwerpen, S. 283-292.

Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G. (2005): JADE Programmer's Guide. http://jade.tilab.com/doc/programmersguide.pdf, zugegriffen am 23.08.2006.

Bellifemine, F.; Caire, G.; Trucco, T.; Rimassa, G. (2006): JADE Administrator's Guide. http://jade.tilab.com/doc/administratorsguide.pdf, zugegriffen am 24.08.2006.

Bellifemine, F.L.; Caire, G.; Greenwood, D. (2007): Developing Multi-Agent Systems with JADE, Wiley & Sons, Chichester 2007.

Benmerzoug, D.; Boufaida, Z.; Boufaida, M. (2004): From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach. Konferenzbeitrag: On the Move to Meaningful Internet Systems 2004: OTM Confederated International Workshops and Posters, GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004, Larnaca, Zypern, S. 495-506.

Berg, M.; Toussaint, P. (2003): The Mantra of Modeling and the Forgotten Powers of Paper: A Sociotechnical View on the Development of Processoriented ICT in Health Care. In: International Journal of Medical Informatics, Vol. 69 (2003) Nr. 2-3, S. 223-234.

Berger, K. (2004): Behandlungspfade als Managementinstrument im Krankenhaus. In: Pfade durch das Klinische Prozessmanagement: Methodik und aktuelle Diskussionen. Hrsg.: Greiling, M. Kohlhammer, Stuttgart 2004, S. 42-64.

Berger, M.; Bauer, B.; Watzke, M. (2001): Towards an Agent-Based Infrastructure for Distributed Virtual Organizations. Konferenzbeitrag: IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises Cambridge, MA, USA, S. 354-355.

Berger, M.; Müller, J.P.; Seitz, C. (2005): Multiagenten-Technologien für Ambient Intelligence. In: it – Information Technology, Vol. 47 (2005) Nr. 1, S. 13-19.

Berger, M.; Rusitschka, S.; Toropov, D.; Watzke, M.; Schlichte, M. (2002): Porting Distributed Agent-Middleware to Small Mobile Devices. Konferenzbeitrag: Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, Bologna, Italien, o.S.

Bergfelder, M.; Nitschke, T.; Sorge, C. (2005): Signaturen durch elektronische Agenten: Vertragsschluss, Form und Beweis. In: Informatik-Spektrum, Vol. 28 (2005) Nr. 3, S. 210-219.

Bergmann, S.; Klinger, A.; Winkler, J. (2006): Agentenbasierte Prozessunterstützung für klinische Prozesse. Interdisziplinäres Projekt, Technische Universität München 2006.

Bernnat, R.; Booz Allen Hamilton GmbH (2006): Endbericht zur Kosten-Nutzen-Analyse der Einrichtung einer Telematik-Infrastruktur im deutschen Gesundheitswesen. http://www.ccc.de/crd/whistleblowerdocs/20060731-Gesundheitstelematik.pdf, zugegriffen am 27.11.2006.

Berwick, D.M. (1996): A Primer on Leading the Improvement of Systems. In: British Medical Journal, Vol. 312 (1996) Nr. 7031, S. 619-622.

Beyer, M.; Kuhn, K.A.; Meiler, C.; Jablonski, S.; Lenz, R. (2004): Towards a Flexible, Process-Oriented IT Architecture for an Integrated Healthcare Network. Konferenzbeitrag: ACM Symposium on Applied Computing, Nicosia, Zypern, S. 264–271.

Beyer, M.; Lenz, R.; Kuhn, K.A. (2006): Health Information Systems. In: it – Information Technology, Vol. 48 (2006) Nr. 1, S. 6-11.

Bhasale, A. (1998): The Wrong Diagnosis: Identifying Causes of Potentially Adverse Events in General Practice Using Incident Monitoring. In: Family Practice, Vol. 15 (1998) Nr. 4, S. 308–318.

Binder, W.; Roth, V. (2002): Secure Mobile Agent Systems Using Java – Where Are We Heading? Konferenzbeitrag: 17th ACM Symposium on Applied Computing, Special Track on Agents, Interactions, Mobility, and Systems, Madrid, Spanien, o.S.

Bird, A.P.; Walji, M.T. (1989): Our Patients Have Access to Their Records. In: British Medical Journal, Vol. 292 (1989) Nr. 6520, S. 595-596.

Birkner, B.R. (2003): Evidenzbasierte Prävention des kolorektalen Karzinoms. In: Deutsche Medizinische Wochenschrift, Vol. 128 (2003), S. 2598-2603.

Bludau, H.; Koop, A.; Herzog, W. (2001): Mobile Computer im Gesundheitswesen: Nützliche Werkzeuge für den Arzt. In: Deutsches Ärzteblatt/PraxisComputer, Vol. 2001 (2001) Nr. 3, S. 22-25.

Bodenreider, O.; Nelson, S.J.; Hole, W.T.; Chang, H.F. (1998): Beyond Synonymy: Exploiting the UMLS Semantics in Mapping Vocabularies. Konferenzbeitrag: AMIA Symposium, S. 815–819.

Boeker, M.; Jentzsch, C.; Klar, R. (2004): Structuring of Medical Documents with XML: Enhancing Retrieval and Presentation of Pathology Reports. Konferenzbeitrag: 11th World Congress on Medical Informatics (MEDINFO 2004), San Francisco, CA, USA, S. 1531.

Bond, A.H.; Gasser, L. (1988): Readings in Distributed Artificial Intelligence, Morgan Kaufmann, San Mateo 1988.

Booch, G. (1994): Object-Oriented Analysis and Design with Applications. (2. Aufl.), Benjamin/Cummings Publishing Company, Inc., Redwood City u.a. 1994.

Bortz, J.; Döring, N. (2002): Forschungsmethoden und Evaluation: Für Human- und Sozialwissenschaftler, Springer, Berlin u.a. 2002.

Bosch, J.; Lundberg, L. (2003): Software Architecture – Engineering Quality Attributes. In: Journal of Systems and Software, Vol. 66 (2003) Nr. 3, S. 183-186.

Bowman, E.H. (1959): The Scheduling-Sequencing Problem. In: Operations Research, (1959) Nr. 7, S. 621-624.

Bratman, M. (1987): Intention, Plans, and Practical Reason, Harvard University Press 1987.

Braubach, L.; Pokahr, A.; Lamersdorf, W. (2004): MedPAge: Rationale Agenten zur Patientensteuerung. In: KI – Zeitschrift für Künstliche Intelligenz, Vol. 4 (2004), S. 34-37.

Brazier, F.M.T.; Catholijn, M.J.; Treur, J. (2002): Principles of Component-based Design of Intelligent Agents. In: Data & Knowledge Engineering, Vol. 41 (2002) Nr. 1, S. 1-27.

Brennan, P.; Safran, C. (2003): Report of Conference Track 3: Patient Empowerment. In: International Journal of Medical Informatics, Vol. 69 (2003) Nr. 2-3, S. 301-304.

Brown, S.L.; Eisenhardt, K.M. (1998): Competing on the Edge: Strategy as Structured Chaos, Harvard Business School Press, Cambridge, MA, USA 1998.

Broy, M. (1998): Informatik: Eine grundlegende Einführung (Band 1: Programmierung und Rechnerstrukturen). (2. Aufl.), Springer, Berlin u.a. 1998.

Broy, M. (2005): Komponenten im Umfeld der elektronischen Gesundheitskarte – Eine Hilfestellung zur Erstellung von Softwarelösungen. http://www.software-offensive-bayern.de/pdf/IntegratedHealthCare.pdf, zugegriffen am 01.03.2006.

Broy, M.; Rausch, A. (2005): Das neue V-Modell XT: Ein anpassbares Modell für Software und System Engineering. In: Informatik-Spektrum, Vol. 28 (2005) Nr. 3, S. 220-229.

Broy, M.; Rumpe, B. (2007): Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. In: Informatik-Spektrum, Vol. 30 (2007) Nr. 1, S. 3-18.

Broy, M.; Stølen, K. (2001): Specification and Development of Interactive Systems, Springer, New York u.a. 2001.

Bruegge, B.; Dutoit, A.H. (2000): Object-Oriented Software-Engineering: Conquering Complex and Changing Systems, Prentice Hall, Upper Saddle River, NJ, USA 2000.

Brunkhorst, I.; Olmedilla, D. (2006): Interoperability for Peer-to-Peer Networks: Opening P2P to the rest of the World. Konferenzbeitrag: European Conference on Technology Enhanced Learning, Kreta, Griechenland, S. 45-60.

Budd, T. (1996): An Introduction to Object-Oriented Programming, Addison-Wesley, Reading, MA, USA 1996.

Bundesamt für Sicherheit in der Informationstechnik (2003): Kommunikations- und Informationstechnik 2010 + 3: Neue Trends und Entwicklungen in Technologie, Anwendungen und Sicherheit, SecuMedia, Ingelheim 2003.

Bundesgesetzblatt im Internet (2003): GMG: Gesetz zur Modernisierung der gesetzlichen Krankenversicherung (GKV-Modernisierungsgesetz – GMG) vom 14. November 2003. http://www.die-gesundheitsreform.de/gesundheitspolitik/gesetze/pdf/gkv-modernisierungsgesetz-gmg.pdf, zugegriffen am 15.12.2005.

Bundesminister der Justiz (1992): Kriterien für die Bewertung von Systemen der Informationstechnik (ITSEC). In: Bundesanzeiger, Vol. 44 (1992) Nr. 147a.

Bundesministerium der Justiz (1988): Sozialgesetzbuch – Fünftes Buch (V) – Gesetzliche Krankenversicherung. http://bundesrecht.juris.de/bundesrecht/sgb_5/gesamt.pdf, zugegriffen am 09.03.2006.

Bundesministerium der Justiz (1990): Bundesdatenschutzgesetz. http://bundesrecht.juris.de/bundesrecht/bdsg_1990/gesamt.pdf, zugegriffen am 08.03.2006.

Bundesrepublik Deutschland (2004): V-Modell XT, Teil 1: Grundlagen des V-Modells. http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/pdf/V-Modell-XT-Teil1.pdf, zugegriffen am 12.02.2007.

Burmeister, B.; Haddadi, G.; Matylis, G. (1997): Applications of Multi-Agent Systems in Traffic and Transportation. In: IEE Proceedings Software Engineering, Vol. 144 (1997) Nr. 1, S. 51-60.

Caire, G. (2003): JADE Tutorial: JADE Programming for Beginners. http://jade.tilab.com/doc/JADEProgramming-Tutorial-for-beginners.pdf, zugegriffen am 23.08.2006.

Caire, G. (2004): The Service-Based Architecture of the New JADE Kernel. http://jade.tilab.com/papers/Jade-the-services-architecture.pdf, zugegriffen am 23.08.2006.

Caire, G. (2006): LEAP User Guide. http://jade.tilab.com/doc/LEAPUserGuide.pdf, zugegriffen am 11.02.2006.

Caire, G.; Coulier, W.; Garijo, M.; Gomez, J.; Pavon, J.; Leal, F.; Chainho, P.; Kearney, P.; Stark, J.; Evans, R.S.; P., M. (2001): Agent Oriented Analysis using MESSAGE/UML. Konferenzbeitrag: Second International Workshop on Agent-Oriented Software Engineering (AOSE), Montreal, Quebec, Canada, S. 119-135.

Caire, G.; Lhuillier, N.; Rimassa, G. (2002): A Communication Protocol for Agents on Handheld Devices. Konferenzbeitrag: Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices in Conjunction with the 2002 Conference on Autonomous Agents and Multiagent Systems, Bologna, Italien, o.S.

Cambliss, M.L.; Conley, J. (1996): Answering Clinical Questions. In: The Journal of Family Practice, Vol. 43 (1996) Nr. 2, S. 140-144.

Camp, R.C. (1989): Benchmarking – The Search for Industry Best Practices that Lead to Superior Performance, Quality Press, Milwaukee, WI, USA 1989.

Camp, R.C. (1995): Business Process Benchmarking, ASQC Press, Milwaukee 1995.

careon.de GmbH (2006): careon-Webseite. www.careon.de, zugegriffen am 07.03.2006.

Caumanns, J. (2006): Der Patient bleibt Herr seiner Daten: Realisierung des eGK-Berechtigungskonzepts über ein ticketbasiertes, virtuelles Dateisystem. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 5, S. 323-331.

Caumanns, J.; Weber, H.; Fellien, A.; Kurrek, H.; Boehm, O.; Neuhaus, J.; Kunsmann, J.; Struif, B. (2006): Die eGK-Lösungsarchitektur: Architektur zur Unterstützung der Anwendungen der elektronischen Gesundheitskarte. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 5, S. 341-348.

Celi, L.A.; Hassan, E.; Marquardt, C.; Breslow, M.; Rosenfeld, B. (2001): The eICU: It's Not Just Telemedicine. In: Critical Care Medicine, Vol. 29 (2001) Nr. 8 (Supplement), S. N183-N189.

Cernuzzi, L.; Rossi, G. (2002): On the Evaluation of Agent Oriented Modelling Methods. Konferenzbeitrag: Object-Oriented Programming, Systems, Languages and Applications 2002 Workshop on Agent Oriented Methodologies, Seattle, WA, USA, S. 21-30.

Chan, Y.E.; Dekker, A.R.; Ramsden, D.J. (2005): Information Systems and Health Care III: Diffusing Health Care Knowledge: A Case Study of the Care Delivery Network. In: Communications of the Association for Information Systems, Vol. 15 (2005) Article 13.

Chavez, A.; Maes, P. (1996): Kasbah: An Agent Marketplace for Buying and Selling Goods. Konferenzbeitrag: First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, S. 75-90.

Chmiel, K.; Gawinecki, M.; Kaczmarek, P.; Szymczak, M.; Paprzycki, M. (2005): Efficiency of JADE Agent Platform. In: Scientific Programming, Vol. 13 (2005) Nr. 2, S. 159-172.

Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S. (2001): Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, zugegriffen am 01.09.2006.

Cimino, J.J. (1997): Intranet Technology in Hospital Information Systems. In: New Technologies in Hospital Information Systems. Hrsg.: Dudeck, J.; Blobel, B.; Lordieck, W.; Bürkle, T. IOS Press, Amsterdam, Berlin 1997, S. 102-109.

Cimino, J.J.; Patel, V.L.; Kushniruk, A.W. (2001): What Do Patients Do with Access to Their Medical Records? Konferenzbeitrag: MEDINFO 2001, London, UK, S. 1440-1444.

Cimino, J.J.; Socratous, S.A.; Clayton, P.D. (1995): Internet as Clinical Information System: Application Development Using the World Wide Web. In: Journal of the American Medical Association, Vol. 2 (1995) Nr. 5, S. 273-284.

Clade, H. (2003): Krankenhäuser: „Lernendes System“. In: Deutsches Ärzteblatt, Vol. 100 (2003) Nr. 42, S. A-2681/B-2237/C-2101.

Clarke, E.M.; Emerson, E.A. (1981): Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. Konferenzbeitrag: Workshop Logics of Programs, o.O., S. 52-71.

Claussen, U. (1993): Objektorientiertes Programmieren, Springer, Berlin, Heidelberg, New York 1993.

CollabNet Inc. (2006): JXTA-Webseite. http://www.jxta.org/, zugegriffen am 24.11.2006.

Committee on Quality of Health Care in America: Institute of Medicine (2001): Crossing the Quality Chasm: A New Health System for the 21st Century, National Academy Press, Washington, D.C., USA 2001.

Corera, J.M.; Laresgoiti, I.; Jennings, N.R. (1996): Using Archon, Part 2: Electricity Transportation Management. In: IEEE Expert, Vol. 11 (1996) Nr. 6, S. 71-79.

Cortese, E.; Quarta, F.; Vitaglione, G. (2002): Scalability and Performance of JADE Message Transport System. Konferenzbeitrag: International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Workshop on AgentCities, Bologna, Italien, o.S.

Crosby, C. (2005): Prozess- und Systemanalyse im Gesundheitswesen und Konzeption einer Agenten basierten Lösung. Masterarbeit, Technische Universität München 2005.

Curtis, B.; Keller, M.I.; Over, J. (1992): Process Modelling. In: Communications of the ACM, Vol. 35 (1992) Nr. 9, S. 75-90.

Czap, H.; Becker, M. (2002): A New Scheduling Approach for Hospital Operating Theatres. In: European Journal of Medical Research, Vol. 5 (2002) Suppl. 1, S. 16-17.

Czap, H.; Becker, M. (2003): Multi-Agent Systems and Microoeconomic Theory: A Negotiation Approach to Solve Scheduling Problems in High Dynamic Environments. Konferenzbeitrag: 36th Annual Hawaii International Conference on System Sciences (HICSS 2003), Big Island, HI, USA, o.S.

Dam, K.H.; Winikoff, M. (2003): Comparing Agent-Oriented Methodologies. Konferenzbeitrag: Autonomous Agents and Multiagent Systems Workshop on Agent-Oriented Information Systems, Melbourne, Australien, S. 78-93.

Davidsson, P.; Johansson, S. (2002): Evaluating Multi-Agent System Architectures: A Case Study Concerning Dynamic Resource Allocation. Konferenzbeitrag: International Workshop on Engineering Societies in the Agents' World, Madrid, Spanien, S. 170-183.

Dawes, K.S. (1972): Survey of General Practice Records. In: British Medical Journal, Vol. 3 (1972) Nr. 820, S. 219-223.

de Bruin, H.; van Vliet, H. (2003): Quality-driven Software Architecture Composition. In: Journal of Systems and Software, Vol. 66 (2003) Nr. 3, S. 269-284.

Decker, K.; Li, J. (1998): Coordinated Hospital Patient Scheduling. Konferenzbeitrag: Third International Conference on Multi-Agent Systems (ICMAS98), Paris, Frankreich, S. 104-111.

DeLoach, S.A.; Wood, M.F.; Sparkman, C.H. (2001): Multiagent Systems Engineering. In: International Journal on Software Engineering and Knowledge Engineering, Vol. 11 (2001) Nr. 3, S. 231-258.

Der Bayerische Landesbeauftragte für den Datenschutz (2005): Datenschutz Bayern: Tätigkeitsbericht 2004 des Bayerischen Landesbeauftragten für den Datenschutz (Kapitel 5: Gesundheitswesen). http://www.datenschutz-bayern.de/tbs/tb21/k5.html, zugegriffen am 28.11.2005.

DeRemer, F.; Kron, H. (1976): Programming-in-the-Large versus Programming-in-the-Small. In: IEEE Transactions on Software Engineering, Vol. 2 (1976) Nr. 2, S. 80-86.

Dijkstra, E.W. (1968): The Structure of "THE"-Multiprogramming System. In: Communications of the ACM, Vol. 11 (1968) Nr. 5, S. 341-346.

Dijkstra, E.W. (1969): The Programming Task Considered as an Intellectual Challenge. Vortrag. Eindhoven, Niederlande: Transkription von McCarthy, D. C.

Dix, A.J. (1998): Human-Computer Interaction. (2. Aufl.), Prentice Hall Europe, London, New York 1998.

Dolin, R.H.; Alschuler, L.; Beebe, C.; Biron, P.V.; Boyer, S.L.; Essin, D.; Kimber, E.; Lincoln, T.; Mattison, J.E. (2001): The HL7 Clinical Document Architecture. In: Journal of the American Medical Informatics Association, Vol. 8 (2001) Nr. 6, S. 552-569.

Druker, M.; Michelbach, T. (2006): Portaltechnologie für den Zugriff von Patienten auf medizinische Daten. Interdisziplinäres Projekt, Technische Universität München 2006.

Eberhardt, C.-T.; Gurzki, T.; Hinderer, H. (2002): Marktübersicht Portal Software für Business-, Enterprise-Portale und E-Collaboration, Fraunhofer IRB, Stuttgart 2002.

Eikel, C.; Delbanco, S. (2003): John M. Eisenberg Patient Safety Awards: The Leapfrog Group for Patient Safety: Rewarding Higher Standards. In: Joint Commission Journal on Quality and Safety, Vol. 29 (2003) Nr. 12, S. 634–639.

Elson, R.B.; Faughnan, J.G.; Connelly, D.P. (1997): An Industrial Process View of Information Delivery to Support Clinical Decision Making: Implications for Systems Design and Process Measures. In: Journal of the American Medical Informatics Association, Vol. 4 (1997) Nr. 4, S. 266-278.

Ely, J.W.; Osheroff, J.A.; Ebell, M.H.; Chambliss, M.L.; Stevermer, J.J.; Vinson, D.C.; Pifer, E.A. (2002): Obstacles to Answering Doctors' Questions about Patient Care with Evidence: Qualitative Study. In: British Medical Journal, Vol. 324 (2002) Nr. 7339, S. 710-716.

Erl, T. (2005): Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, New York 2005.

Essex, B.; Doig, R.; Renshaw, J. (1990): Pilot Study of Records of Shared Care for People with Mental Illnesses. In: British Medical Journal, Vol. 300 (1990) Nr. 6737, S. 1442-1446.

Evans, R.S.; Pestotnik, S.L.; Classen, D.C.; Clemmer, T.P.; Weaver, L.K.; Orme, J.F.J.; Lloyd, J.F.; Burke, J.P. (1998): A Computer-Assisted Management Program for Antibiotics and other Antiinfective Agents. In: The New England Journal of Medicine, Vol. 338 (1998) Nr. 4, S. 232-238.

Eysenbach, G. (2001): What is e-health? In: Journal of Medical Internet Research, Vol. 3 (2001) Nr. 2, S. e20.

Ferrara, F.M. (1997): Healthcare Information Systems Architecture. In: New Technologies in Hospital Information Systems (Band 45). Hrsg.: Dudeck, J.; Blobel, B.; Lordieck, W.; Bürkle, T. IOS Press, Amsterdam u.a. 1997, S. 1-10.

FIPA Modeling TC (2004): Agent UML-Webseite. http://www.auml.org/, zugegriffen am 22.11.2006.

Floyd, C. (1984): A Systematic Look at Prototyping. In: Approaches to Prototyping. Hrsg.: Budde, R.; Kuhlenkamp, K.; Mathiassen, L.; Züllighoven, H. Springer, Berlin u.a. 1984, S. 1-18.

Fowles, J.B.; Kind, A.C.; Craft, C.; Kind, E.A.; Mandel, J.L.; Adlis, S. (2004): Patients' Interest in Reading Their Medical Record: Relation With Clinical and Sociodemographic Characteristics and Patients' Approach to Health Care. In: Archives of Internal Medicine, Vol. 164 (2004) Nr. 7, S. 793-800.

Frandji, B. (1997): Open Architecture for Health Care Systems: The European RICHE Experience. In: New Technologies in Hospital Information Systems (Band 45). Hrsg.: Dudeck, J.; Blobel, B.; Lordieck, W.; Bürkle, T. IOS Press, Amsterdam u.a. 1997, S. 11-23.

Frank, U. (2006): Wissenschaftstheorie der Wirtschaftsinformatik. Gastvortrag am Lehrstuhl für Wirtschaftsinformatik, Technische Universität München.

Frants, V.I.; Shapiro, J.; Voiskunskii, V.G. (1996): Development of IR Systems: New Direction. In: Information Processing & Management, Vol. 32 (1996) Nr. 3, S. 373-386.

Fraunhofer Institut für Biomedizinische Technik (Arbeitsgruppe Medizin-Telematik, 2000): PaDok® – Patientenbegleitende Dokumentation. http://www.ibmt.fraunhofer.de/Produktblaetter/MT_padokdoku_de.pdf, zugegriffen am 29.09.2006.

Freßmann, A.; Maximini, R.; Sauer, T. (2005): Towards Collaborative Agent-Based Knowledge Support for Time-Critical and Business-Critical Processes. Konferenzbeitrag: Professional Knowledge Management: Third Biennial Conference, WM 2005, Kaiserslautern, S. 420-430.

Fridman Noy, N.; Fergerson, R.W.; Musen, M.A. (2000): The Knowledge Model of Protégé 2000: Combining Interoperability and Flexibility. Konferenzbeitrag: EKAW, o.O., S. 17-32.

Fries, J.F. (1974): Alternatives in Medical Record Formats. In: Medical Care, Vol. 12 (1974) Nr. 10, S. 871-881.

Galliers, J.R. (1988): A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict. Diss., University of Cambridge 1988.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995): Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading u.a. 1995.

Gappmair, M.; Häntschel, I. (1997): Die Evaluierung von Workflow-Management-Systemen in Laborstudien. In: Wirtschaftsinformatik – Ergebnisse empirischer Forschung. Hrsg.: Grün, O.; Heinrich, L.J. Springer, Wien, New York 1997, S. 63-77.

Garfield, M.J.; Kamis, A.A.; LeRouge, C.M. (2004): Champion Networks in Federated Interorganizational Systems: Case Studies in Telemedicine. In: Communications of the Association for Information Systems, Vol. 14 (2004) Article 28.

Garrett, L.E.J.; Hammond, W.E.; Stead, W.W. (1986): The Effects of Computerized Medical Records on Provider Efficiency and Quality of Care. In: Methods of Information in Medicine, Vol. 25 (1986) Nr. 3, S. 151-157.

GEHR (2006): The Good European Health Record-Webseite. http://www.chime.ucl.ac.uk/work-areas/ehrs/GEHR/, zugegriffen am 10.05.2006.

Gell-Mann, M. (1995): The Quark and the Jaguar: Adventures in the Simple and Complex, W. H. Freeman, New York 1995.

gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH gematik-Webseite. www.gematik.de, zugegriffen am 2006.

gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH (2006): Speicherplatzbedarf der eGK (Version 1.1.1 vom 07.09.2006). https://www.gematik.de/upload/gematik_eGK_Speicherplatzbedarf_V1_1_1_1556.pdf, zugegriffen am 06.12.2006.

gematik – Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH: Version 1.0.0 vom 28.04.2006 (2006): Spezifikation eHealth-Kartenterminal. http://www.gematik.de/(cjsikkidvzvqju2ngjkjlt55)/upload/gematik_KT_eHealth_Kartenterminal_V1_0_0_621.pdf, zugegriffen am 30.05.2006.

Genesereth, M.R.; Ketchpel, S.P. (1994): Software Agents. In: Communications of the ACM, Vol. 37 (1994) Nr. 7 (July 1994), S. 48-53; 147-148.

Gesakon GmbH (2005): akteonline-Webseite. www.akteonline.de, zugegriffen am 07.03.2006.

Ghosh, S. (2003): Add XML Parsing to Your J2ME Applications. http://www-128.ibm.com/developerworks/library/wi-parsexml/, zugegriffen am 02.01.2007.

Giglio, R.J.; Spears, B.; Rumpf, D.; Eddy, N. (1978): Encouraging Behavior Changes by Use of Client-Held Health Records. In: Medical Care, Vol. 16 (1978) Nr. 9, S. 757-764.

Gmelin, M.; Fischer, H.; Schurr, M.; Schöchlin, J.; Bolz, A. (2001): AVETANA – Integriertes Gesundheitsmanagement für den Patienten. In: Telemedizinführer Deutschland 2002. Hrsg.: Jäckel, A. Deutsches Medizin Forum, Ober-Mörlen 2001, S. 104-105.

Good European Health Record (1995): GEHR Architecture. http://www.chime.ucl.ac.uk/work-areas/ehrs/GEHR/EUCEN/del19.pdf, zugegriffen am 10.10.2006.

Gräber, S. (1996): Communication Services for a Distributed Hospital Information System. In: Methods of Information in Medicine, Vol. 35 (1996) Nr. 3, S. 230-341.

Graf v.d. Schulenburg, J.-M.; Uber, A.; König, W.; Andersen, H.H.; Henke, K.-D.; Laaser, U.; Allhoff, P.G. (1995): Ökonomische Evaluation telemedizinischer Projekte und Anwendungen (Band 22), Nomos Verlagsgesellschaft, Baden-Baden 1995.

Gräslund, K.; Krcmar, H.; Schwabe, G. (1996): The BTOEV Method for Needs-driven Design and Implementation of Telecooperation Systems in Public Administration. Konferenzbeitrag: Information Systems Methodologies 1996: Lessons Learned from the Use of Methodologies, o.O., S. 365-376.

Greenwood, D. (2005): JADE Web Service Integration Gateway (WSIG). http://jade.tilab.com/papers/2005/JADEWorkshopAAMAS/AAMAS05_JADE-Tutorial_WSIG-Slides.pdf, zugegriffen am 11.07.2006.

Griffeth, N.D.; Velthuijsen, H. (1994): The Negotiating Agents Approach to Run-Time Feature Interaction Resolution. Konferenzbeitrag: International Workshop on Feature Interactions in Telecommunications and Software Systems, Amsterdam, Niederlande, S. 217-235.

Grimshaw, J.M.; Russel, J.L. (1998): Effect of Clinical Guidelines on Medical Practice: A Systematic Review of Rigorous Evaluation. In: The Lancet, Vol. 342 (1998) Nr. 8883, S. 1317-1322.

Grimson, W.; Jung, B.; van Mulligen, E.M.; van Ginneken, A.; Pardon, S.; Sottile, P.A. (2002): Extensions to the HISA Standard – the SynEx Computing Environment. In: Methods of Information in Medicine, Vol. 41 (2002) Nr. 5, S. 401-410.

Grütter, R. (2006): Software-Agenten im Semantic Web. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 1, S. 3-13.

Haas, N. (2005a): Konzeption und Implementierung von Sicherheitsdiensten der Telematik-Rahmenarchitektur. Interdisziplinäres Projekt, Technische Universität München 2005a.

Haas, N. (2005b): Konzeption und Implementierung von Sicherheitsdiensten der Telematik-Rahmenarchitektur (Erweiterte Dokumentation zum Protokollieren von Patientenkartenzugriffen). Interdisziplinäres Projekt, Technische Universität München 2005b.

Haas, N. (2005c): Konzeption und Implementierung von Sicherheitsdiensten der Telematik-Rahmenarchitektur (Erweiterte Dokumentation zur implementierten Konnektorfunktionalität). Interdisziplinäres Projekt, Technische Universität München 2005c.

Haas, P. (2005d): Design und Implementierung einer WEB- und CDA-basierten einrichtungsübergreifenden Elektronischen Krankenakte. Konferenzbeitrag: 50. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds), 12. Jahrestagung der Deutschen Arbeitsgemeinschaft für Epidemiologie (dae), Freiburg, S. 91-94.

Habermas, J. (1985): Theorie des kommunikativen Handelns (Band 1: Handlungsrationalität und gesellschaftliche Rationalisierung). (Dritte, durchgesehene Aufl.), Suhrkamp, Frankfurt am Main 1985.

Hasselbring, W. (1997): Federated Integration of Replicated Information Within Hospitals. In: International Journal on Digital Libraries, Vol. 1 (1997) Nr. 3, S. 192-208.

Hasselbring, W. (2006): Software-Architektur. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 1, S. 48-52.

Haux, R. (2005): Health Information Systems – Past, Present, Future. In: International Journal of Medical Informatics, Vol. 75 (2005) Nr. 3-4, S. 268-281.

Haux, R.; Ammenwerth, E.; Herzog, W.; Knaup, P. (2002): Health Care in the Information Society. A Prognosis for the Year 2013. In: International Journal of Medical Informatics, Vol. 66 (2002) Nr. 1-3, S. 3-21.

Hayes-Roth, B.; Hewett, M.; Washington, R.; Hewett, R.; Seiver, A. (1990): Distributing Intelligence within an Individual. In: Distributed Artificial Intelligence (Band 2). Hrsg.: Gasser, L.; Huhns, M.N. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 1990, S. 385-412.

Health Level Seven Inc. Ann Arbor MI (2005): Health Level Seven-Webseite. http://www.hl7.org/, zugegriffen am 15.03.2005.

Heine, C. (2005): Zielorientiertes Requirements Engineering Methoden zur Entwicklung von Agentensystemen. In: it – Information Technology, Vol. 47 (2005) Nr. 1, S. 20-27.

Heine, C.; Herrler, R.; Kirn, S. (2005): ADAPT@AGENT.HOSPITAL: Agent-based Optimization and Management of Clinical Processes. In: International Journal of Intelligent Information Technologies, Vol. 1 (2005) Nr. 1, S. 30-48.

Heine, C.; Herrler, R.; Petsch, M.; Anhalt, C. (2003): ADAPT – Adaptive Multi Agent Process Planning & Coordination of Clinical Trials. Konferenzbeitrag: AMCIS 2003, Tampa, Florida, USA, S. 1832-1834.

Heinen, E. (1976): Grundfragen der entscheidungsorientierten Betriebswirtschaftslehre, Goldmann, München 1976.

Heinrich, L.J.; Heinzl, A.; Roithmayr, F. (2004): Wirtschaftsinformatik-Lexikon. (7., vollst. überarb. u. erw. Aufl.), Oldenbourg, München, Wien 2004.

Heinzl, A.; König, W.; Hack, J. (2001): Erkenntnisziele der Wirtschaftsinformatik in den nächsten drei und zehn Jahren. In: Wirtschaftsinformatik, Vol. 43 (2001) Nr. 3, S. 223-233.

Heitmann, K.U.; Schweiger, R.; Dudeck, J. (2003): Discharge and Referral Data Exchange Using Global Standards – the SCIPHOX Project in Germany. In: International Journal of Medical Informatics, Vol. 70 (2003) Nr. 2-3, S. 195-203.

Heldwein, W.; Rösch, T.; Klose, J.; Riemann, J.F.; Schmitt, W.; Birkner, B.; Hagenmüller, F.; Classen, M. (1999): Leitlinie der Deutschen Gesellschaft für Verdauungs- und Stoffwechselerkrankungen: Endoskopische Terminologie – Ergebnisse eines Konsensusprojektes. http://www.dgvs.de/media/einleitungendos.pdf, zugegriffen am 09.06.2006.

Hellmann, W. (2002): Klinische Pfade: Konzepte, Umsetzung, Erfahrungen, ecomed, Landsberg 2002.

Herrler, R.; Heine, C. (2004): The ADAPT Toolkit-supported Engineering Process for Agent Based Applications. Konferenzbeitrag: Americas Conference on Information Systems (AMCIS 2004), New York City, NY, USA, S. 1794-1805.

Herrler, R.; Heine, C.; Klügl, F. (2002): Appointment Scheduling Among Agents: A Case Study In Designing Suitable Interaction Protocols. Konferenzbeitrag: AMCIS 2002, Dallas, TX, USA, S. 1456-1463.

Hevner, A.R.; March, S.T.; Park, J.; Ram, S. (2004): Design Science in Information Systems Research. In: MIS Quarterly, Vol. 28 (2004) Nr. 1, S. 75-106.

Hillebrand, C. (2006): Portierung einer agentenbasierten elektronischen Patientenakte auf mobile Endgeräte. Diplomarbeit, Technische Universität München 2006.

Hoare, C.A.R.; Wirth, N. (1973): Axiomatic Definition of the Programming Language Pascal. In: Acta Informatica, Vol. 2 (1973), S. 335-355.

Hoffmann, H.; Leimeister, J.M.; Krcmar, H. (2007): Pilotierung mobiler Dienste im Automobilsektor. In: Mobile Dienste im Auto der Zukunft: Konzeption, Entwicklung, Pilotierung (Band 13). Hrsg.: Reichwald, R.; Krcmar, H.; Reindl, S. Josef Eul, Lohmar 2007, S. 125-204.

Holland, J.H. (1998): Emergence: From Chaos to Order, Addison-Wesley, Reading, MA, USA 1998.

Holzinger, A.; Geierhofer, R.; Errath, M. (2007): Semantische Informationsextraktion in medizinischen Informationssystemen. In: Informatik-Spektrum, Vol. 30 (2007) Nr. 2, S. 69-78.

Horn, E.; Schubert, W. (1993): Objektorientierte Software-Konstruktion: Grundlagen – Modelle – Methoden – Beispiele, Carl Hanser, München, Wien 1993.

Humm, B.; Voß, M.; Hess, A. (2006): Regeln für serviceorientierte Architekturen hoher Qualität. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 6, S. 395-411.

Huth, E.J. (1985): Needed: An Economics Approach to Systems for Medical Information. In: Annals of Internal Medicine, Vol. 103 (1985) Nr. 4, S. 617-619.

IEEE Foundation for Intelligent Physical Agents (2006): FIPA-Webseite. http://www.fipa.org/, zugegriffen am 26.04.2006.

Iglesias, C.; Garijo, M.; Gonzales, J.; Velasco, J. (1997): Analysis and Design of Multiagent Systems using MAS-CommonKADS. Konferenzbeitrag: 4th International Workshop on Intelligent Agents IV: Agent Theories, Architectures, and Languages, Providence, RI, USA, S. 313-327.

Imamura, T.; Dillaway, B.; Simon, E. (2002): XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core/, zugegriffen am 14.02.2007.

InterComponentWare AG (2001): Lifesensor – die offene Serviceplattform. In: Telemedizinführer Deutschland 2002. Hrsg.: Jäckel, A. Deutsches Medizin Forum, Ober-Mörlen 2001, S. 294-296.

InterComponentWare AG (2006): LifeSensor-Webseite. http://www.de.lifesensor.com/, zugegriffen am 07.03.2006.

Jablonski, S. (1997): Architektur von Workflow-Management-Systemen. In: Informatik – Forschung und Entwicklung, Vol. 12 (1997) Nr. 2, S. 72-81.

Jablonski, S.; Böhm, M.; Schulze, W. (1997): Workflow-Management: Entwicklung von Anwendungen und Systemen, dpunkt.verlag, Heidelberg 1997.

JADE Board (2005): JADE Security Guide. http://jade.tilab.com/doc/JADE_Security.pdf, zugegriffen am 23.08.2006.

Jaksic, A. (2006): Design und Implementierung eines Bedienkonzeptes für aktive Dokumente. Systementwicklungsprojekt, Technische Universität München 2006.

Janz, B.D.; Pitts, M.G.; Otondo, R.F. (2005): Information Systems and Health Care II: Back to the Future with RFID: Lessons Learned – Some Old, Some New. In: Communications of the Association for Information Systems, Vol. 15 (2005) Article 7.

Jeckle, M.; Zengler, B. (2002): SOAP – aber sicher. In: OBJEKTspektrum, (2002) Nr. 1, S. 17-27.

Jennings, N.R. (2001): An Agent-Based Approach for Building Complex Software Systems. In: Communications of the ACM, Vol. 44 (2001) Nr. 4, S. 35-41.

Jennings, N.R.; Faratin, P.; Johnson, M.J.; Norman, T.J.; O'Brien, P.; Wiegand, M.E. (1996a): Agent-based Business Process Management. In: International Journal of Cooperative Information Systems, Vol. 5 (1996a) Nr. 2-3, S. 105-130.

Jennings, N.R.; Mamdani, E.H.; Corera, J.M.; Laresgoiti, I.; Perriollat, F.; Skarek, P.; Varga, L.Z. (1996b): Using Archon to Develop Real-World DAI Applications, Part 1. In: IEEE Expert, Vol. 11 (1996b) Nr. 6, S. 64-70.

Jennings, N.R.; Sycara, K.; Wooldridge, M. (1998): A Roadmap of Agent Research and Development. In: Autonomous Agents and Multi-Agent Systems, Vol. 1 (1998) Nr. 1, S. 7-38.

Jennings, N.R.; Wooldridge, M.J. (1998): Applications of Intelligent Agents. In: Agent Technology: Foundations, Applications, and Markets. Hrsg.: Jennings, N.R.; Wooldridge, M.J. Springer, Berlin, Heidelberg, New York 1998, S. 3-28.

Junginger, M. (2004): Wertorientierte Steuerung von Risiken im Informationsmanagement. Diss., Universität Hohenheim 2004.

Jürjens, J. (2005): Secure Systems Development with UML, Springer, Berlin, Heidelberg, New York 2005.

Kacprzak, M.; Lomuscio, A.; Penczek, W. (2004): Verification of Multiagent Systems Via Unbounded Model Checking. Konferenzbeitrag: 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York City, NY, USA, S. 638-645.

Kamel Boulos, M.N.; Cai, Q.; Padget, J.A.; Rushton, G. (2006): Using Software Agents to Preserve Individual Health Data Confidentiality in Micro-Scale Geographical Analyses. In: Journal of Biomedical Informatics, Vol. 39 (2006) Nr. 2, S. 160-170.

Kaparthi, S.; Woodlief, N.F. (2005): Information and Health Care VI: Medical Nomograms with Decision Support Systems: Case Study and an Enhanced Architecture. In: Communications of the Association for Information Systems, Vol. 16 (2005) Article 21.

Kaprov, A. (2007): Implementierung eines Bedienkonzepts und einer Web-Schnittstelle für aktive Dokumente. Systementwicklungsprojekt, Technische Universität München 2007.

Kauffmann, S.A. (1995): At Home in the Universe, Oxford University Press, Oxford, UK 1995.

KBV (Kassenärztliche Bundesvereinigung Deutschland) (2007): xDT – Synonym für elektronischen Datenaustausch in der Arztpraxis. http://www.kbv.de/ita/4274.html, zugegriffen am 15.12.2005.

Keller, G.; Meinhardt, S. (1994): SAP R/3-Analyzer: Optimierung von Geschäftsprozessen auf Basis des R/3-Referenzmodells. In: SAP R/3-Analyzer. Hrsg.: SAP AG. SAP PRESS, Walldorf 1994.

Keller, G.; Nüttgens, M.; Scheer, A.-W. (1992): Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)". Universität des Saarlandes, 1992.

Keller, W. (2002): Enterprise Application Integration, dpunkt.verlag, Heidelberg 2002.

Kelly, K. (1994): Out of Control: The Rise of Neo-Biological Civilization, Addison-Wesley, Reading, MA, USA 1994.

Kempa, M.; Mann, Z.A. (2005): Model Driven Architecture. In: Informatik-Spektrum, Vol. 28 (2005) Nr. 4, S. 298-302.

Kephart, J.O.; Chess, D.M. (2003): The Vision of Autonomic Computing. In: Computer, IEEE Computer Society, Vol. 36 (2003) Nr. 1, S. 41-50.

Kern, S.E.; Jaron, D. (2003): Healthcare Technology, Economics and Policy: An Evolving Balance. In: IEEE Engineering in Medicine and Biology, Vol. 22 (2003) Nr. 1, S. 16-19.

Kikic, L.; Ratkovic, M. (2006): Entwicklung einer Java-API für die SAP-Anbindung. Systementwicklungsprojekt, Technische Universität München 2006.

Kim, M.I.; Johnson, K.B. (2002): Personal Health Records: Evaluation of Functionality and Utility. In: Journal of the American Medical Informatics Association, Vol. 9 (2002) Nr. 2, S. 171-180.

Kinny, D.; Georgeff, M.; Rao, A. (1996): A Methodology and Modelling Technique for Systems of BDI Agents. Konferenzbeitrag: Seventh European Workshop on Modelling Autonomous Agents in a MultiAgent World, Eindhoven, Niederlande, S. 56-71.

Kirn, S. (2002): Kooperierende intelligente Softwareagenten. In: Wirtschaftsinformatik, Vol. 44 (2002) Nr. 1, S. 53-63.

Kirn, S. (2005): Agententechnologie. In: it – Information Technology, Vol. 47 (2005) Nr. 1, S. 3-4.

Kirn, S. (2006): Flexibility of Multiagent Systems. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 53-69.

Kirn, S.; Anhalt, C.; Krcmar, H.; Schweiger, A. (2006a): Agent.Hospital – Health Care Applications of Intelligent Agents. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006a, S. 199-220.

Kirn, S.; Heine, C.; Petsch, M.; Puppe, F.; Klügl, F.; Herrler, R. (2000): Agentenorientierte Modellierung vernetzter Logistikkreisläufe als Ausgangspunkt agentenbasierter Simulation – Beispiel Wöchnerinnenstation. Konferenzbeitrag: 2. Kolloquium des DFG-Schwerpunktprogramms „Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien“ (SPP 1083), Ilmenau.

Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. (2006b): Multiagent Engineering, Springer, Berlin, Heidelberg 2006b.

Kleppe, A.; Warmer, J.; Bast, W. (2003): MDA Explained, Addison-Wesley Professional, Boston u.a. 2003.

Klinikum rechts der Isar der Technischen Universität München (2003): Klinikum Rechts der Isar der Technischen Universität München: Jahresbericht 2003. http://www.med.tu-muenchen.de/upload/Jahresbericht-2003.pdf, zugegriffen am 21.02.2006.

Klinikum rechts der Isar der Technischen Universität München (2006): Organigramm des MRI. http://www.med.tu-muenchen.de/de/organisation/struktur/MRI_Organigramm_21.pdf, zugegriffen am 23.06.2006.

Koestler, A. (1984): Die Wurzeln des Zufalls, Scherz, München 1984.

Koop, A.; Matesic, R.; Mösges, R. (2001): Erfahrungen beim Einsatz von Palm-PDAs in einer klinischen Studie. Konferenzbeitrag: 1. Workshop der GMDS-Projektgruppe Mobiles Computing in der Medizin, Köln, S. 45-59.

Kossmann, D.; Leymann, F. (2004): Web Services. In: Informatik-Spektrum, Vol. 27 (2004) Nr. 2, S. 117-128.

Koutkias, V.G.; Chouvarda, I.; Maglaveras, N. (2005): A Multiagent System Enhancing Home-Care Health Services for Chronic Disease Management. In: IEEE Transactions on Information Technology in Biomedicine, Vol. 9 (2005) Nr. 4, S. 528-537.

Krafzig, D.; Banke, K.; Slama, D. (2006): Enterprise SOA: Service-Oriented Architecture Best Practices, Pearson Education, Inc., Upper Saddle River, NJ, USA 2006.

Krasner, G.E.; Pope, S.T. (1988): A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. In: Journal of Object-Oriented Programming, Vol. 1 (1988) Nr. 3, S. 26-49.

Krcmar, H. (2005): Informationsmanagement. (4. Aufl.), Springer, Berlin, Heidelberg, New York 2005.

Krcmar, H.; Horn, E. (1999): Agentensystemarchitekturen für aktive medizinische Dokumente – Informationslogistik in multikontextuellen Domänen (Antrag auf Gewährung einer Sachbeihilfe bei der DFG, Neuantrag). Universität Hohenheim, Universität Potsdam, 1999.

Krcmar, H.; Horn, E. (2001a): Agentensystemarchitekturen für aktive medizinische Dokumente – Informationslogistik in multikontextuellen Domänen (Antrag auf Gewährung einer Sachbeihilfe bei der DFG, Fortsetzungsantrag). Universität Hohenheim, Universität Potsdam, 2001a.

Krcmar, H.; Horn, E. (2001b): Agentensystemarchitekturen für aktive medizinische Dokumente – Informationslogistik in multikontextuellen Domänen (Bericht über die vorliegenden Ergebnisse, Projektetappe 1). Universität Hohenheim, Universität Potsdam, 2001b.

Krcmar, H.; Horn, E. (2003): Agentensystemarchitekturen für aktive medizinische Dokumente – Informationslogistik in multikontextuellen Domänen (Bericht über die vorliegenden Ergebnisse, Projektetappe 2). Technische Universität München, Universität Potsdam, 2003.

Krcmar, H.; Leimeister, J.M.; Klapdor, S.; Hörmann, C. (2007): IT-Management im Krankenhaus: Die Sicht der IT-Entscheider: Eine empirische Studie zur Ermittlung von Herausforderungen und Trends für das IT-Management in deutschen Krankenhäusern. Technische Universität München, Lehrstuhl für Wirtschaftsinformatik, 2007.

Krempels, K.-H.; Nimis, J.; Braubach, L.; Herrler, R.; Pokahr, A. (2003): Entwicklung intelligenter Multi-Multiagentensysteme – Werkzeugunterstützung, Lösungen und offene Fragen. Konferenzbeitrag: Informatik 2003: Innovative Informatikanwendungen, Frankfurt, S. 31-46.

Krüger, I.H. (2000): Distributed System Design with Message Sequence Charts. Diss., Technische Universität München 2000.

Krüger, I.H.; Prenninger, W.; Sandner, R. (2004): Broadcast MSCs. In: Formal Aspects of Computing, Vol. 16 (2004) Nr. 3, S. 194-209.

Kuhn, K.A.; Giuse, D.A. (2001): From Hospital Information Systems to Health Information Systems: Problems, Challenges, Perspectives. In: Methods of Information in Medicine, Vol. 40 (2001) Nr. 4, S. 275-287.

Kuhn, K.A.; Wurst, S.H.R.; Bott, O.J.; Giuse, D.A. (2006): Expanding the Scope of Health Information Systems: Challenges and Developments. In: Methods of Information in Medicine, Vol. 45 (2006) Suppl. 1, S. 43-52.

Kuilboer, M.M.; van der Lei, J.; Bohnen, A.M.; van Bemmel, J.H. (1997): The Availability of Unavailable Information. Konferenzbeitrag: AMIA Annual Fall Symposium, Nashville, TN, USA, S. 749-753.

Kuperman, G.J.; Gibson, R.F. (2003): Computer Physician Order Entry: Benefits, Costs, and Issues. In: Annals of Internal Medicine, Vol. 139 (2003) Nr. 1, S. 31-39.

Kuperman, G.J.; Teich, J.M.; Tanasijevic, M.J.; Ma'Luf, N.; Rittenberg, E.; Jha, A.; Fiskio, J.; Winkelman, J.; Bates, D.W. (1999): Improving Response to Critical Laboratory Results with Automation: Results of a Randomized Controlled Trial. In: Journal of the American Medical Informatics Association, Vol. 6 (1999) Nr. 6, S. 512-522.

Kuziemsky, C.E.; Jahnke, J.H. (2005): Information Systems and Health Care V – A Multimodal Approach to Health Care Decision Support Systems. In: Communications of the Association for Information Systems, Vol. 16 (2005) Article 19.

KVNo Kassenärztliche Vereinigung Nordrhein (2005): D2D: Telematik-Initiative der Kassenärztlichen Vereinigungen. http://kvno.arzt.de/importiert/d2d_konzept.pdf, zugegriffen am 01.09.2006.

Lange, D.B.; Oshima, M. (1999): Seven Good Reasons for Mobile Agents. In: Communications of the ACM, Vol. 42 (1999) Nr. 3, S. 88-89.

Laugharne, R.; Stafford, A. (1996): Access to Records and Client Held Records for People with Mental Illness: A Literature Review. In: Psychiatric Bulletin, (1996) Nr. 20, S. 338-341.

Leape, L.L. (1997): A Systems Analysis Approach to Medical Error. In: Journal of Evaluation in Clinical Practice, Vol. 3 (1997) Nr. 3, S. 213-222.

Leape, L.L.; Bates, D.W.; Cullen, D.J.; Cooper, J.; Demonaco, H.J.; Gallivan, T.; Hallisey, R.; Ives, J.; Laird, N.; Laffel, G.; et al. (1995): Systems Analysis of Adverse Drug Events. In: Journal of the American Medical Association, Vol. 274 (1995) Nr. 1, S. 35-43.

Legner, C.; Heutschi, R. (2007): SOA Adoption in Practice – Findings from Early SOA Implementations. Konferenzbeitrag: European Conference on Information Systems (ECIS 2007), St. Gallen, S. 1643-1654.

Lehmann, T.M. (2005): Handbuch der Medizinischen Informatik. (2. vollständig neu bearbeitete Aufl.), Carl Hanser, München, Wien 2005.

Lehmann, T.M.; Spitzer, K. (2006): Medical Informatics – Challenges of Applied Information Technology in Medicine. In: it – Information Technology, Vol. 48 (2006) Nr. 1, S. 3-5.

Leimeister, J.M.; Krcmar, H.; Horsch, A.; Kuhn, K.A. (2005): Mobile IT-Systeme im Gesundheitswesen, mobile Systeme für Patienten. In: HMD, Praxis der Wirtschaftsinformatik, Vol. 244 (2005) August 2005, S. 74-85.

Leimeister, J.M.; Schweiger, A.; Krcmar, H. (2006): Ortsunabhängiges Management von hochpreisigen mobilen medizinischen Geräten im Krankenhaus auf WLAN-Basis. Konferenzbeitrag: Informatik 2006, Dresden, S. 220-226.

Lenz, R.; Beyer, M.; Meiler, C.; Jablonski, S.; Kuhn, K.A. (2005): Informationsintegration in Gesundheitsversorgungsnetzen: Herausforderungen an die Informatik. In: Informatik-Spektrum, Vol. 28 (2005) Nr. 2, S. 105-119.

Lenz, R.; Elstner, T.; Siegele, H.; Kuhn, K.A. (2002): A Practical Approach to Process Support in Health Information Systems. In: Journal of the American Medical Informatics Association, Vol. 9 (2002) Nr. 6, S. 571–585.

Lenz, R.; Kuhn, K.A. (2001): Intranet Meets Hospital Information Systems: The Solution to the Integration Problem? In: Methods of Information in Medicine, Vol. 40 (2001) Nr. 2, S. 99-105.

Lewin, R. (1992): Complexity: Life at the Edge of Chaos, Macmillian, New York 1992.

Leymann, F.; Altenhuber, W. (1994): Managing Business Processes as an Information Resource. In: IBM Systems Journal, Vol. 33 (1994) Nr. 2, S. 326-348.

Leymann, F.; Roller, D. (1999): Production Workflow: Concepts and Techniques, Prentice Hall PTR, Upper Saddle River, NJ, USA 1999.

Lieberman, H. (1999): Personal Assistants for the Web: An MIT Perspective. In: Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet. Hrsg.: Klusch, M. Springer, Berlin u.a. 1999, S. 279-292.

Linwood, J.; Minter, D. (2004): Building Portals with the Java Portlet API, Apress, Berkeley, CA, USA 2004.

Ljungberg, M.; Lucas, A. (1992): The OASIS Air Traffic Management System. Konferenzbeitrag: Second Pacific Rim International Conference on Artificial Intelligence (PRICAI 1992), Seoul, Korea, o.S.

Lockemann, P.C.; Nimis, J. (2005): Softwareagenten: Von den Eigenschaften zur Architektur. In: it – Information Technology, Vol. 47 (2005) Nr. 1, S. 28-35.

Loos, P.; Scheer, A.-W. (1995): Vom Informationsmodell zum Anwendungssystem – Nutzenpotenziale für den effizienten Einsatz von Informationssystemen. Konferenzbeitrag: Wirtschaftsinformatik 1995, Frankfurt, S. 185-201.

Luck, M.; McBurney, P.; Shehory, O.; Willmott, S. (2004): Agent Technology Roadmap: Overview and Consultation Report. http://www.agentlink.org/roadmap/, zugegriffen am 07.04.2005.

Lundberg, L.; Bosch, J.; Hggander, D.; Bengtsson, P. (1999): Quality Attributes in Software Architecture Design. Konferenzbeitrag: Annual IASTED International Conference Software Engineering and Applications, Scottsdale, AZ, USA, o.S.

Lyyntinen, K.; Lehtinen, E. (1984): On Information Modelling Through Illocutionary Logic. Konferenzbeitrag: Third Scandinavian Research Seminar on Information Modelling and Data Base Management, Tampere, Finnland, S. 35-116.

Ma, P.Y.; Lee, E.Y.S.; Tsuchiya, M. (1982): A Task Allocation Model for Distributed Computing Systems. In: IEEE Transactions on Computers, Vol. C-31 (1982) Nr. 1, S. 41-47.

Mabry, S.L.; Schneringer, T.; Etters, T.; Edwards, N. (2003): Intelligent Agents for Patient Monitoring and Diagnostics. Konferenzbeitrag: ACM Symposium on Applied Computing, Melbourne, FL, USA, S. 257-262.

Maes, P. (1994): Agents That Reduce Work and Information Overload. In: Communications of the ACM, Vol. 37 (1994) Nr. 7, S. 31-40.

Mandelbrot, B. (1999): A Fractal Walk on Wall Street. In: Scientific American, Vol. 280 (1999) Nr. 2, S. 70-71.

Mantzana, V.; Themistocleous, M.; Irani, Z.; Morabito, V. (2007): Identifying Healthcare Actors involved in the Adoption of Information Systems. In: European Journal of Information Systems, (2007) Nr. 16, S. 91-102.

March, S.T.; Smith, G.F. (1995): Design and Natural Science Research on Information Technology. In: Decision Support Systems, Vol. 15 (1995) Nr. 4, S. 251-266.

Masys, D.; Baker, D.; Butros, A.; Cowles, K. (2002): Giving Patients Access to Their Medical Records via the Internet. In: Journal of the American Medical Informatics Association, Vol. 9 (2002) Nr. 2, S. 181-191.

Matthias, L.; Prokosch, H.-U.; Hasselbring, W. (1999): Eine Taxonomie für Kommunikationsserver im Krankenhaus. In: Informatik, Biometrie und Epidemiologie in Medizin und Biologie, Vol. 30 (1999) Nr. 1, S. 21-43.

Maturana, H.R. (1982): Erkennen: Die Organisation und Verkörperung von Wirklichkeit. (2. durchgesehene Aufl.), Vieweg, Wiesbaden 1982.

Matyas, P.; Maurer, U. (2003): Distributed Systems: A History of Middleware. In: Leveraging Business with Web Services: Trend Report 2003. Hrsg.: Dornbusch, P.; Huber, M.; Müller, M.; Landgrebe, J.; Zündt, M. Center for Digital Technology and Management (CDTM), München 2003, S. 94-101.

Mauro, C.; Sunyaev, A.; Leimeister, J.M.; Schweiger, A.; Krcmar, H. (2008): The Clinic Card Solution – A Concept for a Single Sign-On Central Management of Smart Cards in Hospitals. Konferenzbeitrag: 41st Hawaii International Conference on System Sciences (HICSS 2008), Waikoloa, Big Island, HI, USA, o.S.

McDonald, C.J. (1976): Protocol-based Computer Reminders, the Quality of Care and the Nonperfectibility of Man. In: New England Journal of Medicine, Vol. 295 (1976) Nr. 24, S. 1351–1355.

McDonald, C.J. (1997): The Barriers to Electronic Medical Record Systems and How to Overcome Them. In: Journal of the American Medical Informatics Association, Vol. 4 (1997) Nr. 3, S. 213-221.

McDonald, C.J.; Hui, S.L.; Smith, D.M.; Tierney, W.M.; Cohen, S.J.; Weinberger, M. (1984): Reminders to Physicians from an Introspective Computer Medical Record: A Two-Year Randomized Trial. In: Annals of Internal Medicine, Vol. 100 (1984) Nr. 1, S. 130-138.

McDonald, C.J.; Tierney, W.M. (1988): Computer-stored Medical Records: Their Future Role in Medical Practice. In: Journal of the American Medical Association, Vol. 259 (1988) Nr. 23, S. 3433–3440.

McPhee, S.J.; Bird, J.A.; Fordham, D.; Rodnick, J.E.; Osborn, E.H. (1991): Promoting Cancer Prevention Activities by Primary Care Physicians: Results of a Randomized, Controlled Trial. In: Journal of the American Medical Association, Vol. 266 (1991) Nr. 4, S. 538-544.

Mendling, J.; Nüttgens, M. (2004): Transformation of ARIS Markup Language to EPML. Konferenzbeitrag: 3. GI-Workshop „EPK 2004 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten“, Luxemburg, S. 27-38.

Menzies, T.; Pecheur, C. (2005): Verification and Validation and Artificial Intelligence. In: Advances in Computers (Band 65). Hrsg.: Zelkowitz, M. Elsevier, o.O. 2005.

Meyer, B. (2000): Object-Oriented Software Construction. (2. Aufl.), Prentice Hall 2000.

Meyer, H. (2004): Proaktivität und Prozessmanagement in Multiagentensystemen. Großer Beleg, Universität Potsdam 2004.

Microsoft Deutschland GmbH (o.J.-a): Unternehmensbroschüre: Ihre Gesundheit. Unsere Lösungen: Die eHealth Interoperability Plattform, o.J.-a.

Microsoft Deutschland GmbH (o.J.-b): Unternehmensbroschüre: Microsoft Deutschland GmbH Firmenportrait o.J.-b.

Mikkelsen, G.; Aasly, J. (2001): Concordance of Information in Parallel Electronic and Paper Based Patient Records. In: International Journal of Medical Informatics, Vol. 63 (2001) Nr. 3, S. 123-131.

Ministererklärung Brüssel 2003-05-22 (2003): eHealth 2003, May 2003. http://europa.eu.int/information_society/eeurope/ehealth/conference/2003/doc/min_dec_22_may_03.pdf , zugegriffen am 06.10.2007.

Mitra, N. (2003): SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, zugegriffen am 01.09.2006.

Mönch, L. (2006): Autonome und kooperative Steuerung komplexer Produktionsprozesse mit Multi-Agenten-Systemen. In: Wirtschaftsinformatik, Vol. 48 (2006) Nr. 2, S. 107-119.

Moore, C.; Wisnivesky, J.; Williams, S.; McGinn, T. (2001): Medical Errors Related to Discontinuity of Care from an Inpatient to an Outpatient Setting. In: Journal of General Internal Medicine, Vol. 18 (2001) Nr. 8, S. 646-651.

Moore, G.E. (1976): Cramming More Components onto Integrated Circuits. In: Electronics, Vol. 38 (1976) Nr. 8, S. 114-117.

Moreno, A.; Isern, D. (2002): Offering Agent-based Medical Services within the AgentCities Project. Konferenzbeitrag: International Workshop on Agents Applied in Health Care at the 15th European Conference on Artificial Intelligence (ECAI 2002), Lyon, Frankreich, S. 33-39.

Moreno, A.; Isern, D.; Sánchez, D. (2003): Provision of Agent-based Health Care Services. In: AI Communications, Vol. 16 (2003) Nr. 3, S. 167-178.

Morowitz, H.J.; Singer, J.L. (1995): The Mind, the Brain, and Complex Adaptive Systems, Addison-Wesley Publishing, Reading, MA, USA 1995.

Mössenböck, H.-P. (1994): Objektorientierte Programmierung in Oberon-2, Springer, Berlin, Heidelberg, New York 1994.

Mouratidis, H.; Manson, G.; Giorgini, P.; Philp, I. (2002): Modelling an Agent-based Integrated Health and Social Care Information System for Older People. Konferenzbeitrag: International Workshop on Agents Applied in Health Care at the 15th European Conference on Artificial Intelligence (ECAI 2002), Lyon, Frankreich, o.S.

Müller-Schloer, C. (2005): Organic Computing – Systemforschung zwischen Technik und Naturwissenschaften. In: it – Information Technology, Vol. 47 (2005) Nr. 4, S. 179-181.

Müller-Wilken, S. (2001): Mobile Geräte in verteilten Anwendungsumgebungen: Integrationsansatz zwischen Abstraktion und Migration. Diss. 2001.

Nadkarni, P.M.; Marenco, L.; Chen, R.; Skoufos, E.; Shepherd, G.; Miller, P. (1999): Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. In: Journal of the American Medical Informatics Association, Vol. 6 (1999) Nr. 6, S. 478–493.

National Electrical Manufacturers Association (2006a): Digital Imaging and Communication in Medicine (DICOM) Part 1: Introduction and Overview. http://medical.nema.org/dicom/2006/06_01pu.pdf, zugegriffen am 01.09.2006.

National Electrical Manufacturers Association (2006b): Digital Imaging and Communication in Medicine (DICOM) Part 5: Data Structures and Encoding. http://medical.nema.org/dicom/2006/06_05pu.pdf, zugegriffen am 29.09.2006.

National Electrical Manufacturers Association (2006c): Digital Imaging and Communication in Medicine (DICOM) Part 7: Message Exchange. http://medical.nema.org/dicom/2006/06_07pu.pdf, zugegriffen am 01.09.2006.

NEMA (2000): DICOM-Webseite. http://medical.nema.org/, zugegriffen am 15.12.2005.

Neuhaus, J.; Deiters, W.; Wiedeler, M. (2006): Mehrwertdienste im Umfeld der elektronischen Gesundheitskarte: Möglichkeiten und Gestaltung. In: Informatik-Spektrum, Vol. 29 (2006) Nr. 5, S. 332-340.

Nicolescu, V.; Funk, B.; Niemeyer, P.; Heiler, M.; Wittges, H.; Morandell, T.; Visintin, F.; Kleine Stegemann, B. (2006): Entwicklerbuch SAP Exchange Infrastructure. (1. Aufl.), Galileo Press, Bonn 2006.

Niemann, C.; Eymann, T. (2006): Softwareagenten in der Krankenhauslogistik – ein Ansatz zur effizienten Ressourcenallokation. In: HMD, Praxis der Wirtschaftsinformatik, Vol. 251 (2006) Oktober 2006, S. 77-87.

Niemann, H.; Hasselbring, W.; Wendt, T.; Winter, A.; Meiderhofer, M. (2002): Kopplungsstrategien für Anwendungssysteme im Krankenhaus. In: Wirtschaftsinformatik, Vol. 44 (2002) Nr. 5, S. 425-434.

Nimis, J.; Lockemann, P.; Krempels, K.-H.; Buchmann, E.; Böhm, K. (2006): Towards Dependable Agent Systems. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 465-501.

Nitschke, T. (2006): Legal Consequences of Agent Deployment. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 597-618.

Norris, P., R.; Dawant, B.M.; Geissbuhler, A. (1997): Web-Based Data Integration and Annotation in the Intensive Care Unit. Konferenzbeitrag: American Medical Informatics Association Fall Symposium, Nashville, TN, USA, S. 794-798.

Nüttgens, M.; Rump, F.J. (2002): Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). Konferenzbeitrag: Promise 2002 – Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, Potsdam, S. 64-77.

OASIS (2007): OASIS Web Services Business Process Execution Language Version 2.0. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, zugegriffen am 18.07.2007.

Object Management Group (2000): Model Driven Architecture, White Paper, Draft 3.2, November 27, 2000. ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf, zugegriffen am 17.12.2004.

Object Management Group (2001a): Developing in OMG's Model-Driven Architecture, White Paper, November 2001, Revision 2.6. ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf, zugegriffen am 17.12.2004.

Object Management Group (2001b): Model Driven Architecture (MDA), Document number ormsc/2001-07-01. http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01, zugegriffen am 17.12.2004.

Object Management Group (2003): MDA Guide Version, 1.0.1, Document Number omg/2003-06-01. http://omg.org/docs/omg/03-06-01.pdf, zugegriffen am 17.12.2004.

Object Management Group (2007a): Business Modeling & Integration DTF. http://bmi.omg.org/, zugegriffen am 06.06.2007.

Object Management Group (2007b): Catalog of OMG Modeling and Metadata Specifications. http://www.omg.org/technology/documents/modeling_spec_catalog.htm, zugegriffen am 04.08.2007.

Object Management Group (2007c): UML Resource Page. http://www.uml.org/, zugegriffen am 26.02.2007.

Odell, J.; Nodine, M.; Levy, R. (2004): A Metamodel for Agents, Roles, and Groups. Konferenzbeitrag: Agent-Oriented Software Engineering V: 5th International Workshop (AOSE), New York, NY, USA, S. 78-92.

Odell, J.J. (2002): Objects and Agents Compared. In: Journal of Object Technology, Vol. 1 (2002) Nr. 1, S. 41-53.

openEHR (2004): OpenEHR-Webseite. http://www.openehr.org/, zugegriffen am 15.12.2005.

OpenGALEN (1999): OpenGALEN-Webseite. http://www.opengalen.org/, zugegriffen am 15.12.2005.

Organization for the Advancement of Structured Information Standards (2006): ebXML Business Process Specification Schema, Technical Specification v2.0.4. www.ebxml.org, zugegriffen am 06.06.2007.

Ortmann, M.; Böhme, M.; Schweiger, A.; Krcmar, H. (2007): Ermittlung von Verbesserungspotenzialen und Entwurf von Lösungskonzepten zur flexiblen und dynamischen Unterstützung der Prozesse in der Klinik für Anaesthesiologie am Klinikum rechts der Isar der Technischen Universität München durch gezielten IT-Einsatz. Technische Universität München, Fakultät für Informatik, Lehrstuhl für Wirtschaftsinformatik, 2007.

Parunak, H.V.D. (1997): 'Go to the Ant': Engineering Principles from Natural Agent Systems. In: Annals of Operations Research, Vol. 75 (1997), S. 69-101.

Parunak, H.V.D. (1999): Industrial and Practical Applications of Distributed AI. In: Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence. Hrsg.: Weiss, G. MIT Press, Cambridge, MA, USA, London, UK 1999, S. 377-421.

Paulussen, T.O.; Herrler, R.; Hoffmann, A.; Heine, C.; Becker, M.; Franck, M.; Reinke, T.; Strasser, M. (2003a): Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien im Gesundheitswesen. Konferenzbeitrag: Informatik 2003: Innovative Informatikanwendungen, Frankfurt, S. 64-82.

Paulussen, T.O.; Jennings, N.R.; Decker, K.S.; Heinzl, A. (2003b): Distributed Patient Scheduling in Hospitals. Konferenzbeitrag: 18th International Joint Conference on AI, Acapulco, Mexico, S. 1224-1229.

Paulussen, T.O.; Rothlauf, F.; Heinzl, A. (2001): Konzeption eines Koordinationsmechanismus zur dezentralen Ablaufplanung in medizinischen Behandlungspfaden (MedPaCo). Konferenzbeitrag: 5. Internationale Tagung Wirtschaftsinformatik, Augsburg, S. 867-880.

Pedersen, S.; Hasselbring, W. (2004): Interoperabilität für Informationssysteme im Gesundheitswesen auf Basis medizinischer Standards. In: Informatik – Forschung und Entwicklung, Vol. 18 (2004) Nr. 3-4, S. 174-188.

Perriollat, F.; Skarek, P.; Varga, L.Z.; Jennings, N.R. (1996): Using Archon, Part 3: Particle Acceleration Control. In: IEEE Expert, Vol. 11 (1996) Nr. 6, S. 80-86.

Perry, D.E.; Wolf, A.L. (1992): Foundations for the Study of Software Architecture. In: ACM SIGSOFT Software Engineering Notes, Vol. 17 (1992) Nr. 4, S. 40-52.

Pestotnik, S.L.; Classen, D.C.; Evans, R.S.; Burke, J.P. (1996): Implementing Antibiotic Practice Guidelines through Computer-Assisted Decision Support: Clinical and Financial Outcomes. In: Annals of Internal Medicine, Vol. 124 (1996) Nr. 10, S. 884-890.

Peters, J. (2006): Sichere mobile Agenten. In: IT-Sicherheit und Datenschutz – Zeitschrift für rechts- und prüfungssicheres Datenmanagement, Vol. 01/06 (2006), S. 249-253.

Petersen, L.A.; Orav, E.J.; Teich, J.M.; O'Neil, A.C.; Brennan, T.A. (1998): Using a Computerized Sign-out Program to Improve Continuity of Inpatient Care and Prevent Adverse Events. In: The Joint Commission Journal on Quality Improvement, Vol. 24 (1998) Nr. 2, S. 77-87.

Pichler, J.; Plösch, R.; Weinreich, R. (2002): MASIF und FIPA: Standards für Agenten. In: Informatik-Spektrum, Vol. 25 (2002) Nr. 2, S. 91-100.

Pinsdorf, U.; Peters, J.; Hoffmann, M.; Gupta, P. (2002): Context-aware Services Based on Secure Mobile Agents. Konferenzbeitrag: 10th International Conference on Software, Telecommunications & Computer Networked (SoftCOM 2002), Split, Kroatien, S. 366-370.

Pirker, M.; Berger, M.; Watzke, M. (2004): An Approach for FIPA Agent Service Discovery in Mobile Ad-Hoc Environments. Konferenzbeitrag: Autonomous Agents and Multiagent Systems International Workshop on Agents for Ubiquitous Computing, New York City, NY, USA, o.S.

Plsek, P. (2001): Redesigning Health Care with Insights from the Science of Complex Adaptive Systems. In: Crossing the Quality Chasm: A New Health System for the 21st Century. Hrsg.: Committee on Quality of Health Care in America: Institute of Medicine. National Academy Press, Washington, D.C., USA 2001, S. 309-322.

Pokahr, A.; Braubach, L.; Lamersdorf, W. (2005a): Agenten: Technologie für den Mainstream? In: it – Information Technology, Vol. 47 (2005a) Nr. 5, S. 300-307.

Pokahr, A.; Braubach, L.; Lamersdorf, W. (2005b): Jadex: A BDI Reasoning Engine. In: Multi-Agent Programming: Languages, Platforms and Applications. Hrsg.: Bordini, R.; Dastani, M.; Dix, J.; El Fallah-Seghrouchni, A. Springer, Berlin 2005b, S. 149-174.

Popper, K.R.; Eccles, J.C. (1989): Das Ich und sein Gehirn. (8. Neuausg., 1. Aufl. dieser Ausg.), Piper, München 1989.

Porter, M.E.; Teisberg, E.O. (2004): Redefining Competition in Health Care. In: Harvard Business Review, Vol. 82 (2004) Nr. 6, S. 64-76.

Prigogine, I. (1967): Dissipative Structures in Chemical Systems. In: Fast Reactions and Primary Processes in Chemical Kinetics. Hrsg.: Claesson, S. Interscience, New York City, NY, USA 1967.

Prigogine, I. (1980): From Being to Becoming, W. H. Freeman, San Francisco, CA, USA 1980.

Projektgruppe bIT4health (2004a): Erarbeitung einer Strategie zur Einführung der Gesundheitskarte: Solution Outline – Skizzierung der Lösungsarchitektur und Planung der Umsetzung, Version 1.0 vom 09.07.2004. http://www.dimdi.de/dynamic/de/ehealth/karte/downloadcenter/index.html, zugegriffen am 08.03.2006.

Projektgruppe bIT4health (2004b): Erarbeitung einer Strategie zur Einführung der Gesundheitskarte: Telematikrahmenarchitektur für das Gesundheitswesen – Ein Überblick. http://www.dimdi.de/dynamic/de/ehealth/karte/downloadcenter/index.html, zugegriffen am 08.03.2006.

Projektgruppe bIT4health (2004c): Rahmenarchitektur für die Telematikinfrastruktur des Gesundheitswesens – Ein Überblick, Version 1.1 vom 12. August 2004. http://www.dimdi.de/dynamic/de/ehealth/karte/downloadcenter/index.html, zugegriffen am 09.02.2005.

Projektgruppe bIT4health (2004d): Skizzierung der Lösungsarchitektur und Planung der Umsetzung (Solution Outline), Version 1.1 vom 2. Dezember 2004. http://www.dimdi.de/dynamic/de/ehealth/karte/downloadcenter/index.html, zugegriffen am 09.02.2005.

Projektgruppe FuE-Projekt „Lösungsarchitektur" (2005): Spezifikation der Lösungsarchitektur zur Umsetzung der Anwendungen der elektronischen Gesundheitskarte, Version 1.0 vom 14. März 2005:

Erste Fassung der Spezifikation.
http://www.dimdi.de/dynamic/de/ehealth/karte/downloadcenter/index.html, zugegriffen am 14.03.2005.

Prokosch, H.-U. (2001): KAS, KIS, EKA, EPA, EGA, E-Health: Ein Plädoyer gegen die babylonische Begriffsverwirrung in der Medizinischen Informatik. In: Informatik, Biometrie und Epidemiologie in Medizin und Biologie, Vol. 32 (2001) Nr. 4, S. 371-382.

Prokosch, H.-U.; Ganslandt, T.; Dumitru, R.C.; Ückert, F. (2006): Telemedicine and Collaborative Health Information Systems. In: it – Information Technology, Vol. 48 (2006) Nr. 1, S. 12-23.

Pyper, C.; Amery, J.; Watson, M.; Crook, C. (2004): Access to Electronic Health Records in Primary Care – A Survey of Patients' Views. In: Medical Science Monitor, Vol. 10 (2004) Nr. 11, S. SR17-SR22.

Rahm, E.; Bernstein, P.A. (2001): A Survey of Approaches to Automatic Schema Matching. In: The VLDB Journal, (2001) Nr. 10, S. 334-350.

Ranganathan, S.R. (1931): The Five Laws of Library Science, Madras Library Association (Madras, Indien), Edward Goldston (London, UK) 1931.

Rao, A.S.; Georgeff, M.P. (1995): BDI Agents: From Theory to Practice. Konferenzbeitrag: First International Conference on Multiagent Systems, San Francisco, CA, USA, S. 312-319.

Rechenberg, P.; Pomberger, G. (1997): Informatik-Handbuch, Carl Hanser, München, Wien 1997.

Rechenzentrum des Klinikum rechts der Isar der TU München (2003): Rahmenkonzept für ein Klinikinformationskonzept im Klinikum rechts der Isar, 2003.

Rector, A.L.; Nowlan, W.A. (1994): The GALEN Project. In: Computer Methods and Programs in Biomedicine, Vol. 45 (1994) Nr. 1-2, S. 75-78.

Red Hat Middleware LLC (2006): Relational Persistence for Java and .NET. http://www.hibernate.org/, zugegriffen am 04.08.2006.

Regenstrief Institute Inc. (2004a): Logical Observation Identifiers Names and Codes (LOINC). http://www.regenstrief.org/loinc/, zugegriffen am 03.01.2005.

Regenstrief Institute Inc. (2004b): LOINC Background. http://www.regenstrief.org/loinc/background/, zugegriffen am 29.09.2006.

Reichertz, P.L. (1979): Das Medizinische System Hannover: Erreichtes und Erfahrendes. Konferenzbeitrag: 22. Jahrestagung der GMDS (Informationsverarbeitung in der Medizin: Wege und Irrwege), S. 40-61.

Reinke, T. (2003): Architekturbasierte Konstruktion von Multiagentensystemen. Diss., Universität Potsdam 2003.

Reinke, T.; Horn, E.; Hoffmann, A.; Krcmar, H. (2002): Bridging the Gap between Domain Analysis and Software Models for MAS – Initial Results and Further Studies, Vortrag zum 6. Kolloquium des DFG-Schwerpunktprogramms „Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien" (SPP 1083), Aachen.

Reinke, T.; Horn, E.; Hoffmann, A.; Krcmar, H. (2003): Aktive medizinische Dokumente – Domänenspezifische Anforderungen und technische Realisierung als offenes Framework, Vortrag zum 7. Kolloquium des DFG-Schwerpunktprogramms „Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien" (SPP 1083), Potsdam.

Richter, J.-P.; Haller, H.; Schrey, P. (2005): Serviceorientierte Architektur. In: Informatik-Spektrum, Vol. 28 (2005) Nr. 5, S. 413-416.

Riddle, W.E. (1984): Advancing the State of the Art in Software System Prototyping. In: Approaches to Prototyping. Hrsg.: Budde, R.; Kuhlenkamp, K.; Mathiassen, L.; Züllighoven, H. Springer, Berlin u.a. 1984, S. 19-26.

Riedemann, E.H. (1997): Testmethoden für sequentielle und nebenläufige Software-Systeme, B. G. Teubner, Stuttgart 1997.

Riedl, M. (2007): Scheduling auf der Basis von Software-Agenten im klinischen Umfeld. Diplomarbeit, Technische Universität München 2007.

Rieger, W. (1928): Einführung in die Privatwirtschaftslehre, Krische, Nürnberg 1928.

Rind, D.M.; Safran, C.; Philips, R.S.; Wang, Q.; Calkins, D.R.; Delbanco, T.L.; Bleich, H.L.; Slack, W.V. (1994): Effect of Computer-Based Alerts on the Treatment and Outcomes of Hospitalized Patients. In: Archives of Internal Medicine, Vol. 154 (1994) Nr. 13, S. 1511-1517.

Rochner, F.; Müller-Schloer, C. (2005): Emergence in Technical Systems. In: it – Information Technology, Vol. 47 (2005) Nr. 4, S. 195-200.

Roland Berger & Partner GmbH – International Management Consultants (1997): Telematik im Gesundheitswesen: Perspektiven der Telemedizin in Deutschland. Im Auftrag des Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie und in Zusammenarbeit mit dem Bundesministerium für Gesundheit, 1997.

Romm, F.J.; Putnam, S.M. (1981): The Validity of the Medical Record. In: Medical Care, Vol. 19 (1981) Nr. 3, S. 310-315.

Rose, T.; Sedlmayr, M.; Knublauch, H.; Friesdorf, W. (2006): Agent-Based Information Logistics. In: Multiagent Engineering: Theory and Applications in Enterprises. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 239-254.

Rosenfeld, B.A.; Dorman, T.; Breslow, M.J.; Pronovost, P.; Jenckes, M.; Zhang, N.; Anderson, G.; Rubin, H. (2000): Intensive Care Unit Telemedicine: Alternate Paradigm for Providing Continuous Intensivist Care. In: Critical Care Medicine, Vol. 28 (2000) Nr. 12, S. 3925-3931.

Rosenschein, J.S.; Genesereth, M.R. (1985): Deals Among Rational Agents. Konferenzbeitrag: Ninth International Joint Conference on Artificial Intelligence (IJCAI), Los Angeles, CA, USA, S. 91-99.

Ross, S.E.; Todd, J.; Moore, L.A.; Beaty, B.L.; Wittevrongel, L.; Lin, C.-T. (2005): Expectations of Patients and Physicians Regarding Patient-accessible Medical Records. In: Journal of Medical Internet Research, Vol. 7 (2005) Nr. 2, Article e13.

Roth, V. (2002): Empowering Mobile Software Agents. Konferenzbeitrag: 6th IEEE Mobile Agents Conference, Barcelona, Spanien, S. 47-63.

Roth, V.; Jalali, M. (2001): Concepts and Architecture of a Security-centric Mobile Agent Server. Konferenzbeitrag: 5th International Symposium on Autonomous Decentralized Systems (ISADS 2001), Dallas, TX, USA, S. 435-442.

Rügge, I. (2004): Zwischen- und Abschlussbericht zur Maßnahme „Mobile Anwendungen im Gesundheitswesen“: [wearLab] im Technologie-Zentrum Informatik. Universität Bremen, 2004.

Sachverständigenrat für die Konzertierte Aktion im Gesundheitswesen (2003): Gutachten 2003: Finanzierung, Nutzerorientierung und Qualität. http://dip.bundestag.de/btd/15/005/1500530.pdf, zugegriffen am 15.12.2005.

Sackmann, S.; Eymann, T.; Müller, G. (2002): EMIKA – Real-Time Controlled Mobile Information Systems in Health Care Applications. Konferenzbeitrag: Mobiles Computing in der Medizin, Heidelberg, S. 151-158.

Sarnikar, S.; Zhao, J.L.; Gupta, A. (2005): Medical Information Filtering Using Rule-based and Content-based Approaches. Konferenzbeitrag: Americas Conference on Information Systems (AMCIS 2005), Omaha, NE, USA, o.S.

Sauer, C. (1999): Deciding the Future for IS Failures: Not the Choice You Might Think. In: Rethinking Management Information Systems. Hrsg.: Currie, W.; Galliers, B. Oxford University Press, New York, NY, USA 1999, S. 279–309.

Schelp, J.; Schwinn, A. (2005): Extending the Business Engineering Framework for Application Integration Purposes. Konferenzbeitrag: ACM Symposium on Applied Computing, Santa Fe, NM, USA, S. 1333-1337.

Scherrer, J.-R.; Spahni, S. (1999): Healthcare Information System Architecture (HISA) and its Middleware Models. Konferenzbeitrag: AMIA Annual Symposium, Washington, D.C., USA, S. 935-939.

Schmalenbach, E. (1911-1912): Die Privatwirtschaftslehre als Kunstlehre. In: Zeitschrift für Handelswissenschaftliche Forschung, Vol. 6 (1911-1912), S. 304-320.

Schmatz, K.-D. (2004): Java 2 Micro Edition: Entwicklung mobiler Anwendungen mit CLCD und MIDP, dpunkt, Heidelberg 2004.

Schmidt, B.; Urban, C. (1999): Operative Planung und Steuerung von Gesundheitsdienstleistungen im Krankenhaus: Der Einsatz von PECS-Agenten im medizinischen Bereich (Internes Arbeitspaper). Universität Passau, 1999.

Schneider, J.H. (2001): Online Personal Medical Records: Are They Reliable for Acute/Critical Care? In: Critical Care Medicine, Vol. 29 (2001) Nr. 8, S. 196-201.

Schneier, B. (1996): Applied Cryptography, John Wiley & Sons, Inc., New York u.a. 1996.

Schoenberg, R.; Safran, C. (2000): Internet Based Repository of Medical Records that Retains Patient Confidentiality. In: British Medical Journal, Vol. 321 (2000) Nr. 7270, S. 1199-1203.

Schoop, M. (1998a): A Language-Action Perspective on Cooperative Documentation Systems – Habermas and Searle in Hospital. Konferenzbeitrag: Third International Workshop on the Language Action Perspective on Communication Modelling, Jönköping, Schweden, S. 1-11.

Schoop, M. (1998b): Theoretical Foundations for Cooperative Documentation Systems. Konferenzbeitrag: 43. Tagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS e. V.), Bremen, S. 218-221.

Schoop, M. (1999): An Empirical Study of Multidisciplinary Communication in Healthcare using a Language-Action Perspective. Konferenzbeitrag: Fourth International Workshop on the Language Action Perspective on Communication Modelling (LAP 99), Kopenhagen, Dänemark, S. 59-72.

Schoop, M. (2001): Cooperative Document Management in Multidisciplinary Healthcare. In: Text Databases & Document Management: Theory and Practice. Hrsg.: Chin, A.G. Idea Group Publishing, Hershey et al. 2001, S. 160-222.

Schoop, M.; Wastell, D.G. (1999): Effective Multidisciplinary Communication in Healthcare: Cooperative Documentation Systems. In: Methods of Information in Medicine, Vol. 38 (1999) Nr. 4-5, S. 265-271.

Schreiber, G.; Akkermans, H.; Anjewierden, A.; de Hoog, R.; Shadbolt, N.; Van de Velde, W.; Wielinga, B. (1999): Knowledge Engineering and Management: The CommonKADS Methodology, MIT Press, Cambridge, MA, USA 1999.

Schubert, C.; Zarnekow, R.; Brenner, W. (1998): A Methodology for Classifying Intelligent Software Agents. Konferenzbeitrag: European Conference on Information Systems (ECIS 1998), Aix-en-Provence, Frankreich, S. 304-316.

Schwabe, G. (2001): Bedarfsanalyse. In: CSCW-Kompendium: Lehrbuch zum computerunterstützten kooperativen Arbeiten. Hrsg.: Schwabe, G.; Streitz, N.A.; Unland, R. Springer, Berlin u.a. 2001, S. 361-372.

Schwabe, G.; Hertweck, D.; Krcmar, H. (1997): Partizipation und Kontext bei der Erstellung einer Telekooperationsumgebung: Erfahrungen aus dem Projekt CUPARLA. Konferenzbeitrag: Informatik 1997: Informatik als Innovationsmotor, Aachen, S. 370-379.

Schwabe, G.; Krcmar, H. (1996a): Darstellung des Needs Driven Approach. In: Verbundprojekt BTÖV: Bedarf für Telekooperation in öffentlichen Verwaltungen (Band 3: Die BTÖV-Methode). Hrsg. Universität Hohenheim, Stuttgart 1996a, S. 230-241.

Schwabe, G.; Krcmar, H. (1996b): Der Needs Driven Approach: Eine Methode zur bedarfsgerechten Gestaltung von Telekooperation. Konferenzbeitrag: D-CSCW, Hohenheim, S. 69-88.

Schwarze, J.-C.; Teßmann, S.; Sassenberg, C.; Müller, M.; Prokosch, H.-U.; Ückert, F. (2005a): Eine modulare Elektronische Gesundheitsakte als Plattform einer verteilten Entwicklung. Konferenzbeitrag: 50. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (gmds), 12. Jahrestagung der Deutschen Arbeitsgemeinschaft für Epidemiologie (dae), Freiburg, S. 114-116.

Schwarze, J.-C.; Tessmann, S.; Sassenberg, C.; Müller, M.; Prokosch, H.-U.; Ückert, F. (2005b): Eine modulare Gesundheitsakte als Antwort auf Kommunikationsprobleme im Gesundheitswesen. In: Wirtschaftsinformatik, Vol. 47 (2005b) Nr. 3, S. 187-195.

Schweiger, A.; Bastian, T.; Krcmar, H. (2005): Agentenbasierte elektronische Patientenakten. Konferenzbeitrag: 5. Workshop der GMDS-Projektgruppe Mobiles Computing in der Medizin, Freiburg, S. 45-59.

Schweiger, A.; Hillebrand, C.; Sunyaev, A.; Krcmar, H. (2006a): Portierung einer agentenbasierten elektronischen Patientenakte auf mobile Endgeräte. Konferenzbeitrag: Mobiles Computing in der Medizin, Frankfurt, S. 28-45.

Schweiger, A.; Krcmar, H. (2004): Multi-Agent Systems for Active, Dynamic Activity Areas. Konferenzbeitrag: Americas Conference on Information Systems (AMCIS 2004), New York City, NY, USA, S. 242-245.

Schweiger, A.; Krcmar, H. (2005): Designing Multi-Agent Systems – The NDA-Approach Applied in Health Care. Konferenzbeitrag: Americas Conference on Information Systems (AMCIS 2005), Omaha, NE, USA, S. 1011-1019.

Schweiger, A.; Krcmar, H. (2006): Active, Medical Documents in Health Care. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 301-318.

Schweiger, A.; Leimeister, J.M.; Krcmar, H. (2005a): Auf dem Weg zur integrierten Versorgung im Gesundheitswesen: Teil 1. In: Krankenhaus-IT Journal, Vol. 2005 (2005a) Nr. 1, S. 62-64.

Schweiger, A.; Leimeister, J.M.; Krcmar, H. (2005b): Computer Integrated Manufacturing – Übertragung auf die integrierte Versorgung im Gesundheitswesen: Teil 2. In: Krankenhaus-IT Journal, Vol. 2005 (2005b) Nr. 2, S. 96-98.

Schweiger, A.; Leimeister, J.M.; Krcmar, H. (2007): Auf dem Weg zur integrierten Versorgung im Gesundheitswesen am Beispiel Krankenhaus: Industrie-Parallelen aus Sicht der Wirtschaftsinformatik. In: Wissensmanagement im Krankenhaus: Effizienz- und Qualitätssteigerungen durch

versorgungsorientierte Organisation von Wissen und Prozessen. Hrsg.: Bohnet-Joschko, S. Deutscher Universitäts-Verlag, Wiesbaden 2007, S. 97-110.

Schweiger, A.; Leimeister, J.M.; Niggemann, J.; Feußner, H.; Krcmar, H. (2006b): Softwareagenten für die Überwindung von Medienbrüchen bei der Patientenversorgung – ein Fallbeispiel aus dem Klinikum rechts der Isar der Technischen Universität München. In: HMD, Praxis der Wirtschaftsinformatik, Vol. 251 (2006b) Nr. Oktober 2006, S. 88-100.

Schweiger, A.; Sunyaev, A.; Leimeister, J.M.; Krcmar, H. (2007a): Information Systems and Healthcare XX: Toward Seamless Healthcare with Software Agents. In: Communications of the Association for Information Systems, Vol. 19 (2007a) Article 33, S. 692-709.

Schweiger, A.; Sunyaev, A.; Mauro, C.; Leimeister, J.M.; Krcmar, H. (2007b): Single Sign-On Clinic Card-Lösung – Ein Konzept zur zentralen Verwaltung von Gesundheitskarten im stationären Umfeld. Konferenzbeitrag: Informatik 2007: Informatik trifft Logistik, Bremen, S. 463-468.

Scriven, M. (1980): The Logic of Evaluation, Edgepress, Inverness, CA, USA 1980.

Searle, J.R. (1969): Speech Acts: An Essay in the Philosophy of Language, Cambridge University Press, Cambridge, UK 1969.

Shabot, M.M.; Lobue, M.; Chen, J. (2000): Wireless Clinical Alerts for Physiologic, Laboratory and Medication Data. Konferenzbeitrag: American Medical Informatics Association 2000 Annual Symposium, Los Angeles, CA, USA, S. 789-793.

Shaw, M.; Garlan, D. (1996): Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, Inc., Upper Saddle River, NJ, USA 1996.

Shehory, O. (2000): Software Architecture Attributes of Multi-agent Systems. Konferenzbeitrag: International Workshop on Agent-Oriented Software Engineering, Limerick, Irland, S. 195-200.

Shehory, O.; Sturm, A. (2001): Evaluation of Modelling Techniques for Agent-Based Systems. Konferenzbeitrag: 5th International Conference on Autonomous Agents, Montreal, Quebec, Canada, S. 624-631.

Shneiderman, B.; Plaisant, C. (2005): Designing the User Interface: Strategies for Effective Human-Computer Interaction. (4. Aufl.), Addison-Wesley, Boston, MA, USA 2005.

Simon, H. (1996): The Sciences of the Artificial. (3. Aufl.), The MIT Press, Cambridge, MA, USA 1996.

Simoneit, M. (1998): Informationsmanagement in Universitätsklinika: Konzeption und Implementierung eines objektorientierten Referenzmodells, duv Gabler Edition Wissenschaft, Wiesbaden 1998.

Sittig, D.F. (2001): Personal Health Records on the Internet: A Snapshot of the Pioneers at the End of the 20th Century. In: International Journal of Medical Informatics, Vol. 65 (2001) Nr. 1, S. 1-6.

Slavtchev, V. (2007): Portaltechnologie für die Bereitstellung von Patientendaten. Interdisziplinäres Projekt, Technische Universität München 2007.

Smith, H.A.; McKeen, J.D. (2006): IT in 2020: The Next Frontier. In: MIS Quarterly Executive, Vol. 5 (2006) Nr. 3, S. 125-136.

Smith, R. (1996): What Clinical Information Do Doctors Need? In: British Medical Journal, Vol. 313 (1996) Nr. 7064, S. 1062-1068.

SNOMED International (2005): SNOMED International Web-Seite. http://www.snomed.org/index.html, zugegriffen am 15.12.2005.

SNOMED International (o.J.-a): SNOMED CT Brochure. http://www.snomed.org/news/pdfs/CTbrochure0902.pdf, zugegriffen am 22.03.2006.

SNOMED International (o.J.-b): SNOMED CT Components. http://www.snomed.org/snomedct/documents/SNOMED_CT_Components_000.pdf, zugegriffen am 22.03.2006.

Spreckelsen, C.; Lethen, C.; Heeskens, I.; Spitzer, K. (2001): Mobile Entscheidungsunterstützung für pädiatrische Medikation. Konferenzbeitrag: 1. Workshop der GMDS-Projektgruppe Mobiles Computing in der Medizin, Köln, S. 72-82.

Spreckelsen, C.; Liem, S.; Winter, C.; Spitzer, K. (2006): Cognitive Tools for Medical Knowledge Management. In: it – Information Technology, Vol. 48 (2006) Nr. 1, S. 33-43.

Stacey, R.D. (1996): Complexity and Creativity in Organizations, Berrett-Koehler, San Francisco, CA, USA 1996.

Steinmetz, R.; Wehrle, K. (2004): Peer-to-Peer-Networking & -Computing. In: Informatik-Spektrum, Vol. 27 (2004) Nr. 1, S. 51-54.

Sterritt, R. (2005): Autonomic Computing. In: Innovations in Systems and Software Engineering, Vol. 1 (2005) Nr. 1, S. 79-88.

Stoermer, C.; Bachmann, F.; Verhoef, C. (2003): SACAM: The Software Architecture Comparison Analysis Method (CMU/SEI-2003-TR-006, ESC-TR-2003-006). Carnegie Mellon Software Engineering Institute, 2003.

Strüver, S.-C. (2006): Standardbasiertes EAI-Vorgehen am Beispiel des Investment Bankings. Diss., Technische Universität Berlin 2006.

Stumpe, J.; Orb, J. (2005): SAP Exchange Infrastructure. (1. Aufl.), Galileo Press GmbH, Bonn 2005.

Sun Microsystems Inc. (2002): Datasheet Java 2 Platform, Micro Edition. http://www.sun.com/aboutsun/media/presskits/ctia2004/J2ME.pdf, zugegriffen am 29.12.2006.

Sun Microsystems Inc. (2003): Java Portlet Specification, Version 1.0. http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html, zugegriffen am 11.01.2007.

Sunyaev, A. (2005): Telematik im Gesundheitswesen – Sicherheitsaspekte. Diplomarbeit, Technische Universität München 2005.

Sunyaev, A.; Leimeister, J.M.; Schweiger, A.; Krcmar, H. (2006): Integrationsarchitekturen für das Krankenhaus – Status quo und Zukunftsperspektiven. In: Information Management & Consulting, Vol. 21 (2006) Nr. 1, S. 28-35.

Sunyaev, A.; Leimeister, J.M.; Schweiger, A.; Krcmar, H. (2007): Die elektronische Gesundheitskarte und die Sicherheit: Ein Vorschlag zur entwicklungsbegleitenden Sicherheitsevaluation aus Anwendersicht. Konferenzbeitrag: Informatik 2007: Informatik trifft Logistik, Bremen, S. 469-474.

Sunyaev, A.; Leimeister, J.M.; Schweiger, A.; Krcmar, H. (2008): IT-Standards and Standardization Approaches in Healthcare. In: Encyclopedia of Healthcare Information Systems. Hrsg.: Wickramasinghe, N.; Geisler, E. Idea Group Reference, o.O. 2008, o.S.

Sutherland, J.; van den Heuvel, W.-J. (2002): Enterprise Application Integration and Complex Adaptive Systems. In: Communications of the ACM, Vol. 45 (2002) Nr. 10, S. 59-64.

Sycara, K. (1999): In-Context Information Management through Adaptive Collaboration of Intelligent Agents. In: Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet. Hrsg.: Klusch, M. Springer, Berlin u.a. 1999, S. 78-99.

Sycara, K.P. (1998): Multiagent Systems. In: AI Magazine, Vol. 19 (1998) Nr. 2, S. 79-92.

Szyperski, C. (2002): Component Software: Beyond Object-Oriented Programming. (2. Aufl.), Addison-Wesley, London u.a. 2002.

Tan, J.; Wen, J.H.; Awad, N. (2005): Health Care and Services Delivery Systems as Complex Adaptive Systems. In: Communications of the ACM, Vol. 48 (2005) Nr. 5, S. 36-44.

Tanenbaum, A.S. (2003): Computer Networks. (4. Aufl.), Prentice Hall, Upper Saddle River, NJ, USA 2003.

Tang, P.C.; Fafchamps, D.; Shortliffe, E.H. (1994): Traditional Medical Records as a Source of Clinical Data in the Outpatient Setting. Konferenzbeitrag: Annual Symposium on Computer Applications in Medical Care, Washington, D.C., USA, S. 575–579.

Tang, P.C.; LaRosa, M.P.; Gorden, S.M. (1999): Use of Computer-based Records, Completeness of Documentation, and Appropriateness of Documented Clinical Decisions. In: Journal of the American Medical Informatics Association, Vol. 6 (1999) Nr. 3, S. 245–251.

Taranovych, Y.; Rudolph, S.; Förster, C.; Krcmar, H. (2006): Bedarfsgerechte Entwicklung von webbasierten Projekt-Coaching-Plattformen am Beispiel der WebCo@ch-Plattform. In: Webbasiertes Projekt-Coaching. Hrsg.: Krcmar, H. Josef Eul, Lohmar 2006, S. 221-256.

Taranovych, Y.; Schermann, M.; Schweiger, A.; Krcmar, H. (2007): Enhancing Portal Design. In: Encyclopaedia of Portal Technology and Applications (Band 1). Hrsg.: Tatnall, A. Information Science Reference, Hershey, PA, USA, New York City, NY, USA 2007, S. 353-359.

Tate, K.E.; Gardner, R.M.; Scherting, K. (1995): Nurses, Pagers, and Patient-Specific Criteria: Three Keys to Improved Critical Value Reporting. Konferenzbeitrag: Annual Symposium on Computer Applications in Medical Care, New Orleans, LA, USA, S. 164-168.

Tate, K.E.; Gardner, R.M.; Waever, L.K. (1990): A Computerized Laboratory Alerting System. In: MD Computing, Vol. 7 (1990) Nr. 5, S. 296-301.

Telecom Italia Lab (2006): JADE-Webseite. http://jade.tilab.com/, zugegriffen am 28.02.2006.

TeleTrusT Deutschland e.V. (2006): SICCT Secure Interoperable ChipCardTerminal, Version 1.03. http://gematik.de/upload/SICCT_Spezifikation_103_511.pdf, zugegriffen am 26.02.2007.

Tennenhouse, D. (2000): Proactive Computing. In: Communications of the ACM, Vol. 43 (2000) Nr. 5, S. 43-50.

Thaller, G.G. (2002): Software-Test: Verifikation und Validation. (2., aktualis. u. erw. Aufl.), Heinz Heise GmbH & Co KG, Hannover 2002.

The Foundation for Intelligent Physical Agents (2002): FIPA SL Content Language Specification. http://www.fipa.org/specs/fipa00008/SC00008I.pdf, zugegriffen am 20.11.2006.

The Foundation for Intelligent Physical Agents (FIPA, 1998): FIPA 98 Specification Part 1, Agent Management. http://www.fipa.org/specs/fipa00002/OC00002.pdf, zugegriffen am 30.11.2005.

The Foundation for Intelligent Physical Agents (FIPA, 2002): FIPA SL Content Language Specification. http://www.fipa.org/specs/fipa00008/, zugegriffen am 30.09.2007.

The Foundation for Intelligent Physical Agents (FIPA, 2005): FIPA Specifications. http://www.fipa.org/specifications/index.html, zugegriffen am 06.07.2005.

The openEHR Foundation (2005): Introducing openEHR. http://svn.openehr.org/specification/TRUNK/publishing/openEHR/introducing_openEHR.pdf, zugegriffen am 01.09.2006.

Timm, I.J.; Scholz, T.; Fürstenau, H. (2006): From Testing to Theorem Proving. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 531-554.

Torres da Silva, V.; De Lucena, C.J.P. (2007): Modelling Multi-Agent Systems. In: Communications of the ACM, Vol. 50 (2007) Nr. 5, S. 103-108.

Tufo, H.M.; Speidel, J.J. (1971): Problems with Medical Records. In: Medical Care, Vol. 9 (1971) Nr. 6, S. 509-517.

Ückert, F.; Goerz, M.; Ataian, M.; Tessmann, S.; Prokosch, H.-U. (2003): Empowerment of Patients and Communication with Health Care Professionals through an Electronic Health Record. In: International Journal of Medical Informatics, Vol. 70 (2003) Nr. 2-3, S. 99-108.

Ückert, F.; Görz, M.; Ataian, M.; Prokosch, H.-U. (2002): akteonline – An Electronic Healthcare Record as a Medium for Information and Communication. Konferenzbeitrag: Medical Informatics Europe, Budapest, Ungarn, S. 293-297.

Ückert, F.; Müller, M. (2004): Eine elektronische Gesundheitsakte zur Unterstützung von Patienten und Institutionen des Gesundheitswesens. In: Forum der Medizin-Dokumentation und Medizin-Informatik, Vol. 5 (2004) Nr. 4, S. 100-104.

United Nations Economic Commission for Europe (2005): UN/EDIFACT-Webseite. http://www.unece.org/trade/untdid/welcome.htm, zugegriffen am 15.12.2005.

United States National Library of Medicine (2004): About the UMLS Resources. http://www.nlm.nih.gov/research/umls/about_umls.html, zugegriffen am 15.12.2005.

United States National Library of Medicine (2005): Unified Medical Language System-Webseite. http://www.nlm.nih.gov/research/umls/, zugegriffen am 15.12.2005.

van Aart, C.; Pels, R.; Caire, G.; Bergenti, F. (2002): Creating and Using Ontologies for Agent Communication. Konferenzbeitrag: Workshop on Ontologies in Agent Systems at the 1st Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Bologna, Italien, o.S.

van Walraven, C.; Weinberg, A.L. (1995): Quality Assessment of a Discharge Summary System. In: Canadian Medical Association Journal, Vol. 152 (1995) Nr. 9, S. 1437–1442.

Varela, F.; Coutinho, A. (1991): A Second Generation Immune Networks. In: Immonology Today, Vol. 12 (1991) Nr. 5, S. 159-166.

Varian, H.R. (1994): Mikroökonomie. (3., völlig überarb. u. stark erw. Aufl.), R. Oldenbourg, München, Wien 1994.

Varshney, U. (2005): Pervasive Healthcare: Applications, Challenges and Wireless Solutions. In: Communications of the Association for Information Systems, Vol. 16 (2005), S. 57-72.

VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) (2006a): VCS – der Kommunikationsstandard: Allgemeine Informationen. http://www.vdap.de/index.php?nav=104, zugegriffen am 29.09.2006.

VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) (2006b): VCS – der Kommunikationsstandard: Technische Erläuterungen. http://www.vdap.de/index.php?nav=99, zugegriffen am 29.09.2006.

VDAP (Verband Deutscher Arztinformationssystemhersteller und Provider e.V.) (2006c): VCS – der Kommunikationsstandard: Wer kann VCS nutzen? http://www.vdap.de/index.php?nav=100, zugegriffen am 29.09.2006.

VDAP (Verband Deutscher Arztpraxis-Softwarehersteller e.V. (2006): VCS – Der neue Standard für die elektronische Arzt-Arzt-Kommunikation. www.vdap.de, zugegriffen am 06.12.2006.

Vegoda, P. (2002): Introducing the IHE (Integrating the Healthcare Enterprise) Concept. In: Journal of Healthcare Information Management, Vol. 16 (2002) Nr. 1, S. 22-24.

Vera, A. (2006): Strategische Allianzen im deutschen Krankenhauswesen: Ein empirischer Vergleich von horizontalen und vertikalen Kooperationen. In: Zeitschrift für Betriebswirtschaft, Vol. 76 (2006) Nr. 9, S. 835-865.

Vermeulen, I.; Bohte, S.; Somefun, K.; La Poutré, H. (2006): Improving Patient Activity Schedules by Multi-agent Pareto Appointment Exchanging. Konferenzbeitrag: IEEE Joint Conference on E-Commerce Technology (CEC 2006) and Enterprise Computing, E-Commerce and E-Services (EEE 2006), San Francisco, CA, USA, S. 56-63.

Volpp, K.G.; Grande, D. (2003): Residents' Suggestions for Reducing Errors in Teaching Hospitals. In: New England Journal of Medicine, Vol. 348 (2003) Nr. 9, S. 851–855.

von Eiff, W.; Ziegenbein, R. (2003): Entwicklung von Prozeßmodellen im Krankenhaus. In: Geschäftsprozeßmanagement: Methoden und Techniken für das Management von Leistungsprozessen im Krankenhaus (Band 4). Hrsg.: von Eiff, W.; Ziegenbein, R., 2. (Aufl.). Verlag Bertelsmann Stiftung, Gütersloh 2003, S. 55-82.

Waegemann, C.P. (1999): Current Status of EPR Development in the US. Konferenzbeitrag: Toward An Electronic Health Record Europe, London, UK, S. 116-118.

Waldrop, M.M. (1992): Complexity: The Emerging Science at the Edge of Order and Chaos, Simon and Schuster, New York City, NY, USA 1992.

Walker, J.; Köberle, D.; Strasser, F. (2005): e-MOSAIC: Monitoring von Symptomen, Lebensqualität und Clinical Benefit. Konferenzbeitrag: 5. Workshop der GMDS-Projektgruppe Mobiles Computing in der Medizin, Freiburg, S. 25-30.

Wang, C.; Ohe, K. (1999): A CORBA-Based Object Framework with Patient Identification Translation and Dynamic Linking. In: Methods of Information in Medicine, Vol. 38 (1999) Nr. 1, S. 56-65.

Wavish, P.; Graham, M. (1996): A Situated Action Approach to Implementing Characters in Computer Games. In: International Journal of Applied Artificial Intelligence, Vol. 10 (1996) Nr. 1, S. 53-74.

Weikum, G.; Vossen, G. (2002): Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery, Morgan Kaufmann Publishers, San Francisco, CA, USA 2002.

Weiser, M. (1991): The Computer for the 21st Century. In: Scientific American, (1991) Sept. 1991, S. 94-110.

Weiß, G. (2002): Agent Orientation in Software Engineering. In: The Knowledge Engineering Review, Vol. 16 (2002) Nr. 4, S. 349-373.

Wheatley, M.J. (1992): Leadership and New Science: Learning about Organization from an Orderly Universe, Berrett-Koehler, San Francisco, CA, USA 1992.

Whiting-O'Keffe, Q.E.; Simborg, D.W.; Epstein, W.V.; Warger, A. (1985): A Computerized Summary Medical Record System Can Provide More Information than the Standard Medical Record. In: Journal of the American Medical Association, Vol. 254 (1985) Nr. 9, S. 1185-92.

Wiesman, F.; Hasman, A.; Braun, L.; van den Herik, J. (2006): Information Retrieval in Medicine: The Visual and the Invisible. In: it – Information Technology, Vol. 48 (2006) Nr. 1, S. 24-32.

Wijnstra, J.G. (2003): From Problem to Solution with Quality Attributes and Design Aspects. In: Journal of Systems and Software, Vol. 66 (2003) Nr. 3, S. 199-211.

Wilczek, S. (2007): Aktive elektronische Dokumente in Telekooperationsumgebungen – Konzept und Einsatzmöglichkeiten am Beispiel elektronischer Patientenakte. Diss., Universität Hohenheim 2007.

Willard, K.E.; Connelly, D.P.; Johnson, J.R. (1996): Radical Improvements in the Display of Clinical Microbiology Results: A Web-based Clinical Information System. In: The American Journal of Medicine, Vol. 101 (1996) Nr. 5, S. 541-549.

Wilson, E.O. (1971): The Insect Societies, Harvard University Press, Cambridge, MA, USA 1971.

Wilson, R.M.; Harrison, B.T.; Gibberd, R.W.; Hamilton, J.D. (1999): An Analysis of the Causes of Adverse Events from the Quality in Australian Health Care Study. In: The Medical Journal of Australia, Vol. 170 (1999) Nr. 9, S. 411-415.

Wilson, S.; Ruscoe, W.; Chapman, M.; Miller, R. (2001): General Practitioner-Hospital Communications: A Review of Discharge Summaries. In: Journal of Quality in Clinical Practice, Vol. 21 (2001) Nr. 4, S. 104-108.

Winter, A.F.; Ammenwerth, E.; Bott, O.J.; Brigl, B.; Buchauer, A.; Gräber, S.; Grant, A.; Häber, A.; Hasselbring, W.; Haux, R.; Heinrich, A.; Janssen, H.; Kock, I.; Penger, O.-S.; Prokosch, H.-U.; Terstappen, A.; Winter, A. (2001): Strategic Information Management Plans: The Basis for Systematic Information Management in Hospitals. In: International Journal of Medical Informatics, Vol. 64 (2001) Nr. 2-3, S. 99-109.

Winter, A.F.; Winter, A.; Becker, K.; Bott, O.; Brigl, B.; Gräber, S.; Hasselbring, W.; Haux, R.; Jostes, C.; Penger, O.-S.; Prokosch, H.-U.; Ritter, J.; Schütte, R.; Terstappen, A. (1999): Referenzmodelle

für die Unterstützung des Managements von Krankenhausinformationssystemen. In: Informatik, Biometrie und Epidemiologie in Medizin und Biologie, Vol. 30 (1999) Nr. 4, S. 173-189.

Winter, A.F.; Zimmerling, R.; Bott, O.J.; Gräber, S.; Haas, P.; Hasselbring, W.; Haux, R.; Heinrich, A.; Jaeger, R.; Kock, I.; Möller, D.P.F.; Penger, O.-S.; Prokosch, H.-U.; Ritter, J.; Terstappen, A.; Winter, A. (1998): Das Management von Krankenhausinformationssystemen: Eine Begriffsdefinition. In: Informatik, Biometrie und Epidemiologie in Medizin und Biologie, Vol. 29 (1998) Nr. 2, S. 93-105.

Wissenschaftliche Kommission der Wirtschaftsinformatik (WKWI, 1994): Profil der Wirtschaftsinformatik: Ausführungen der Wissenschaftlichen Kommission der Wirtschaftsinformatik. In: Wirtschaftsinformatik, Vol. 36 (1994) Nr. 1, S. 80-81.

Woelk, P.-O.; Rudzio, H.; Zimmermann, R.; Nimis, J. (2006): Agent.Enterprise in a Nutshell. In: Multiagent Engineering: Theory and Applications in Enterprises. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 73-90.

Wood, M.F.; DeLoach, S.A. (2000): An Overview of the Multiagent Systems Engineering Methodology. Konferenzbeitrag: First International Workshop on Agent-Oriented Software Engineering (AOSE), Limerick, Ireland, S. 207-221.

Wooldridge, M. (1997): Agent-Based Software Engineering. In: Software Engineering, IEEE Proceedings, Vol. 144 (1997) Nr. 1, S. 26-37.

Wooldridge, M. (2002): An Introduction to MultiAgent Systems, John Wiley & Sons, Ltd., Chichester, UK 2002.

Wooldridge, M.; Jennings, N.R. (1995): Intelligent Agents: Theory and Practice. In: The Knowledge Engineering Review, Vol. 10 (1995) Nr. 2, S. 115-152.

Wooldridge, M.; Jennings, N.R.; Kinny, D. (1999): A Methodology for Agent-Oriented Analysis and Design. Konferenzbeitrag: Third International Conference on Autonomous Agents (Agents 99), Seattle, WA, USA, S. 69-75.

Wooldridge, M.; Jennings, N.R.; Kinny, D. (2000): The Gaia Methodology for Agent-Oriented Analysis and Design. In: Autonomous Agents and Multi-Agent Systems, Vol. 3 (2000) Nr. 4, S. 285-312.

Workflow Management Coalition (WfMC, 2005): Workflow Management Coalition Workflow Standard: Process Definition Interface – XML Process Definition Language (XPDL), Version 2.00. http://www.wfmc.org/, zugegriffen am 06.06.2007.

World Health Organization (2006): International Classification of Diseases (ICD). http://www.who.int/classifications/icd/en/, zugegriffen am 29.09.2006.

Wyatt, J.C.; Wright, P. (1998): Design Should Help Use of Patients' Data. In: The Lancet, Vol. 352 (1998) Nr. 9137, S. 1375–1378.

Yan, Y.C.; Lundstrom, S.F. (1989): The Post-Game Analysis Framework: Developing Resource Management Strategies for Concurrent Systems. In: IEEE Transactions on Knowledge and Data Engineering, Vol. 1 (1989) Nr. 3, S. 293-309.

Yu, E. (2001a): Agent-Oriented Modelling: Software Versus the World. Konferenzbeitrag: Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001, Montreal, Quebec, Kanada, S. 206-225.

Yu, E. (2001b): Agent Orientation as a Modelling Paradigm. In: Wirtschaftsinformatik, Vol. 43 (2001b) Nr. 2, S. 123-132.

Yu, E.; Cysneiros, L.M. (2002): Agent-Oriented Methodologies – Towards A Challenge Exemplar. Konferenzbeitrag: 4th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002 at CAiSE 2002), Toronto, Kanada, o.S.

Zachewitz, L. (2004a): Konsistenzsicherung in agentenbasierten Informationssystemen. Konferenzbeitrag: 16. GI-Workshop über Grundlagen von Datenbanken, Heinrich-Heine Universität Düsseldorf, Monheim, S. 118-122.

Zachewitz, L. (2004b): MEDUSA: A Multiagent System for Establishing Electronic Healthcare Records. Konferenzbeitrag: 16th European Conference on Artificial Intelligence (ECAI 2004) – Workshop 7: Second Workshop on Agents Applied in Health Care, Valencia, Spanien, S. 31-37.

Zachewitz, L.; Schwolow, A. (2004): MEDUSA: Agentensystem zur intra- und interinstitutionellen Zusammenführung von Patienteninformationen. Konferenzbeitrag: 11. Interdisziplinärer Workshop KIS/RIS/PACS, Schloss Rauischholzhausen, Gießen, o.S.

Zang, S.; Hofer, A.; Otmar, A. (2004): Cross-Enterprise Business Process Management Architecture – Methods and Tools for Flexible Collaboration. Konferenzbeitrag: On the Move to Meaningful Internet Systems 2004: OTM Confederated International Workshops and Posters, GADA, JTRES, MIOS, WORM, WOSE, PhDS, and INTEROP 2004, Larnaca, Zypern, S. 483-494.

Zendler, A. (1997): Strukturierung, Änderung und Verteilung von objektorientierten Anwendungen. Diss., Universität Potsdam 1997.

Zimmerman, J. (1978): Physician Utilization of Medical Records: Preliminary Determinations. In: Medical Informatics, Vol. 3 (1978) Nr. 1, S. 27-5.

Zimmermann, B.J.; Lindberg, C.; Plsek, P.E. (1998): Edgeware: Insights from Complexity Science for Health Care Leaders, VHA Publishing, Dallas, TX 1998.

Zmud, R. (1997): Editor's Comments. In: MIS Quarterly, Vol. 21 (1997) Nr. 2, S. xxi-xxii.

Zöller, A.; Rothlauf, F.; Paulussen, T.O.; Heinzl, A. (2006): Benchmarking of Multiagent Systems. In: Multiagent Engineering. Hrsg.: Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. Springer, Berlin, Heidelberg, New York 2006, S. 557-574.

Zuckerman, J.; Stafield, B.; Hochreiter, C.; Kovasznay, B. (1975): Validating the Content of Pediatric Outpatient Medical Records by Means of Tape-Recording Doctor-Patient Encounters. In: Pediatrics, Vol. 56 (1975) Nr. 3, S. 407-411.

Zweben, S.H.; Edwards, S.H.; Weide, B.W.; Hollingsworth, J.E. (1995): The Effects of Layering and Encapsulation on Software Development Cost and Quality. In: IEEE Transactions on Software Engineering, Vol. 21 (1995) Nr. 3, S. 200-208.

Anhang

A.1 Komponentenorientierung mit Java

Im Java-Umfeld unterscheidet *Szyperski* die folgenden Komponentenmodelle (2002, 302-304):

- *Applets*: Einbettung eines Java-Programms in den Web Browser
- *Java™ Servlets*: Komponente zur Verarbeitung von Browser-Anfragen auf einem Server
- *JavaBeans*: Zusammenfassung von Klassen und Ressourcen in einer Komponente mit Container-Unterstützung, einsetzbar auf Clients und Servern. Container dienen hier lediglich zur Einkapselung der JavaBeans.
- *Enterprise JavaBeans*: Die Funktionalität des Containers wird durch deklarative Deployment-Deskriptoren seiner Beans beschrieben.
- *Application Client Components*: Java-Applikationen, die auf einem Client laufen und EJBs sowie Ressourcen eines J2EE-Servers verwenden.

Das genannte JavaBeans-Modell besitzt im Detail die folgenden Charakteristika (Szyperski 2002, 285):

- *Ereignisse*: JavaBeans können als Quellen oder Senken von Ereignissen fungieren.
- *Eigenschaften*: Eigenschaften eines JavaBeans werden durch set- und get-Methoden bearbeitet.
- *Introspektion*: Identifikation der Eigenschaften, Ereignisse und Methoden, die ein JavaBean besitzt
- *Customization*: Durch das Setzen der Eigenschaften kann ein JavaBean individualisiert werden.
- *Persistenz*: JavaBeans können für einen späteren Aufruf persistent gemacht werden.

Die Differenzierung zwischen JavaBeans und Enterprise JavaBeans kann nicht durch die Aufteilung der Laufzeitumgebung Client oder Server vorgenommen werden (Szyperski 2002, 303). JavaBeans zeichnen sich nach *Szyperski* (2002, 308) vielmehr durch die folgenden Eigenschaften aus: Das Kommunikationsmodell von JavaBeans basiert auf verbindungsorientierter Programmierung, welche durch eine Verbindung über Ereignisquellen und -senken induziert wird. Dabei lehnt sich das Programmiermodell nah an die Objektorientierung an, weil z.B. Komponenten wie Objekte durch Instanzbildung generiert werden, wenn diese benötigt werden. Das JavaBeans Container-Modell erlaubt eine hierarchische Zusammenstellung von JavaBeans über Subsysteme. Weiterhin wird durch die Einführung eines InfoBus, über den die Kommunikation erfolgt, die Entkopplung zwischen JavaBeans erreicht.

Im Gegensatz zu JavaBeans besitzen EJBs die folgenden Eigenschaften (Szyperski 2002, 308-311): EJBs sind nicht verbindungsorientiert. EJBs zeichnet die kontextuelle Komposition besonders aus. Darunter wird die automatische Komposition von Komponenteninstanzen mit geeigneten Diensten und Ressourcen verstanden. Diese Dienste werden über Deployment-De-

skriptoren beschrieben und können automatisch gefunden werden. Unter dieser kontextuellen Komposition kann eine Umhüllung der Funktionalitäten der Komponenten durch den EJB-Container verstanden werden. Diese Container werden durch EJB-Server zur Verfügung gestellt. Damit können die Funktionalitäten der EJBs nicht direkt aufgerufen werden, sondern werden über den Container mediiert. Die Schnittstellen der Container werden entfernten Clients über Stub-Objekte zur Verfügung gestellt, wobei die Kommunikation z.B. über Java RMI erfolgt.

Hinsichtlich Interoperabilität mit anderen Komponentenmodellen kann festgestellt werden, dass EJBs vollständig in eine CORBA-Umgebung eingebettet werden können (Szyperski 2002, 247). Die Beschreibung der Komponentenorientierung mit CORBA ist Gegenstand des folgenden Abschnittes A.2.

A.2 Komponentenorientierung mit CORBA

Das CORBA Component Model (CCM) erweitert EJBs logisch (Szyperski 2002, 247), indem verbindungsorientierte Programmierung unterstützt wird (Szyperski 2002, 308). Eine CCM-Anwendung ist dabei aus mehreren CCM-Komponenten (siehe Abbildung A.2-1) zusammengesetzt und wird im Folgenden beschrieben (Szyperski 2002, 248-252): XML-Dateien beschreiben die Details einer CCM-Komponente bzw. für einen Komponentenverband die zugehörigen CCM-Komponenten und ihre Deployment-Konfiguration. Eine CCM-Komponente wiederum besteht aus mehreren Segmenten. Applikationen werden zur Laufzeit auf der Granularität dieser Segmente geladen. Eine Komponente besitzt wohl definierte Schnittstellen zur Spezifikation der angebotenen und benötigten Dienste sowie Kanäle für ein- und ausgehende Ereignisse. Jede Komponente befindet sich in einem CCM-Container, welcher eine nach außen sichtbare Schnittstelle zur Spezifikation der Schnittstellen für die benötigten Dienste besitzt.

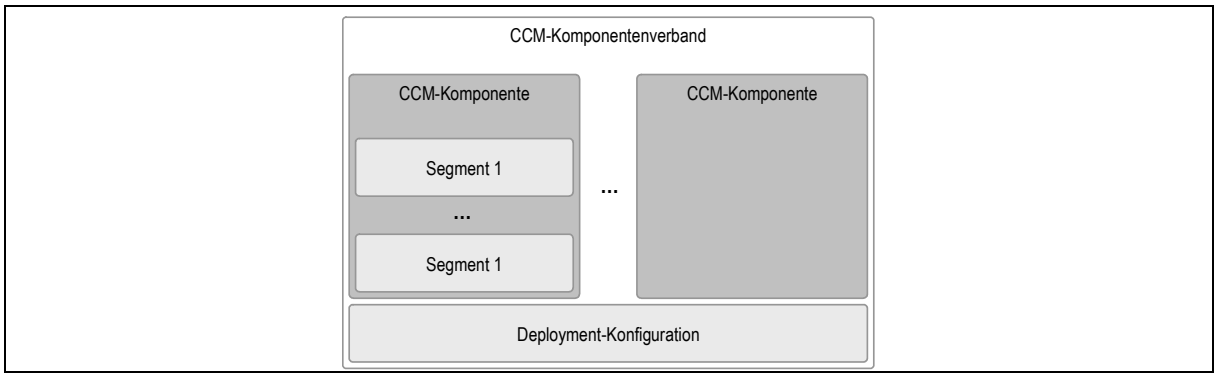

Abbildung A.2-1: ***Komponentenverband, Komponenten und Segmente im CCM***
Quelle: In Anlehnung an *Szyperski* (2002, 250)

Aus programmiertechnischer Sicht besitzt eine CCM-Komponente (siehe Abbildung A.2-2) die folgenden Elemente (Szyperski 2002, 250-251):

- *Facets*: Bereitstellung einer Schnittstelle für einen angebotenen Dienst

- *Receptacles*: Spezifikation einer Schnittstelle für benötigte Dienste über die Relation zu einem Facet Stub einer geeigneten Komponenteninstanz
- *Event Sources* und *Event Sinks*: Verbindung zu Ereigniskanälen zur Kommunikation von Ereignissen
- *Primärschlüssel*: Eindeutige Identifikation einer Komponenteninstanz
- *Attribute*: Der Zugriff auf Attribute erfolgt über definierte Schnittstellen.
- *Konfiguration*: Die initiale Konfiguration wird durch get- und set-Operationen ermöglicht.
- *Equivalent Interface*: Die Navigation zwischen den Facets wird durch ein Equivalent Interface realisiert.
- *Home Interface*: Diese Schnittstelle einer Komponente (nicht einer Komponenteninstanz) erlaubt die Generierung von Komponenteninstanzen.

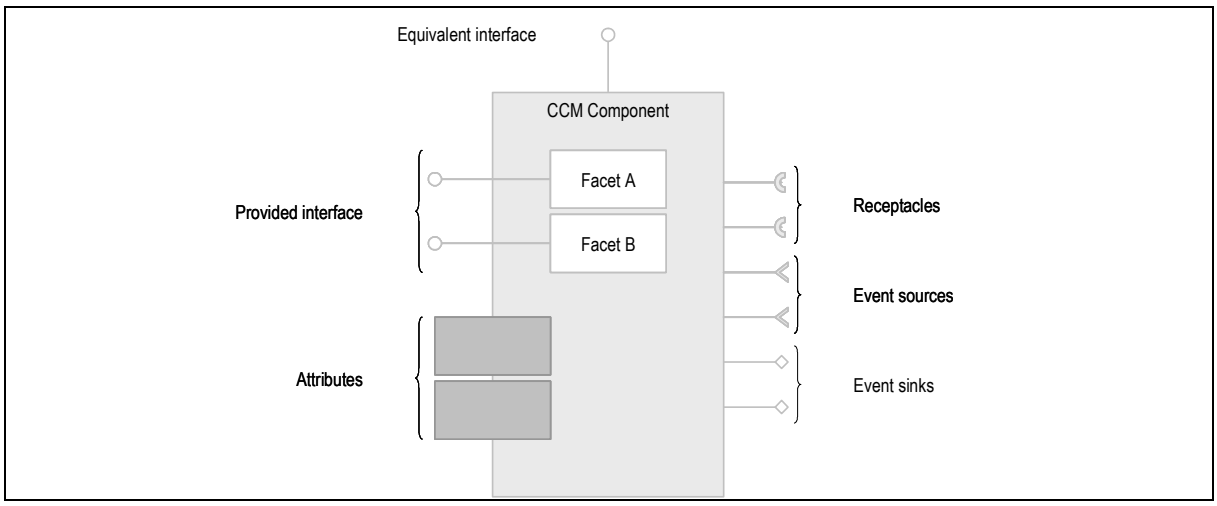

Abbildung A.2-2: ***CCM-Komponente***
Quelle: In Anlehnung an *Szyperski* (2002, 251)

A.3 Komponentenorientierung mit COM und CLR

Die Stufen der Komponentenorientierung bei Microsoft umfassen die Technologien COM, DCOM, COM+ und CLR, welche im Folgenden in Anlehnung an die Ausführungen bei *Szyperski* (2002, 330-335, 343-345, 356-357 und 357-362) beschrieben werden:

In dem binären Standard COM (siehe Abbildung A.3-1) werden als wesentliche Elemente Zeiger auf Schnittstellenknoten in der Form von Schnittstellen definiert. Ein COM-Objekt kann dabei mehrere solcher Schnittstellen besitzen. Die Schnittstellenknoten zeigen auf eine Liste von Funktionsreferenzen. Objekte innerhalb eines COM-Objektes können die zur Verfügung gestellten Schnittstellen implementieren. In Abbildung A.3-1 werden bspw. die Schnittstellen A und B durch Object 1 implementiert. Die gestrichelten Pfeile in Abbildung A.3-1 definieren die Navigation zwischen den Schnittstellenknoten, womit von jedem Knoten aus jeder andere erreicht werden kann.

Jedes COM-Objekt besitzt die Schnittstelle IUnknown. Darüber werden die Operationen QueryInterface, AddRef und Release implementiert, deren Funktionalitäten im Folgenden beschrieben werden: Über die Operation QueryInterface kann an ein COM-Objekt die Anfrage gerichtet werden, ob der übergebene Schnittstellenname als Schnittstelle von diesem COM-Objekt unterstützt wird.

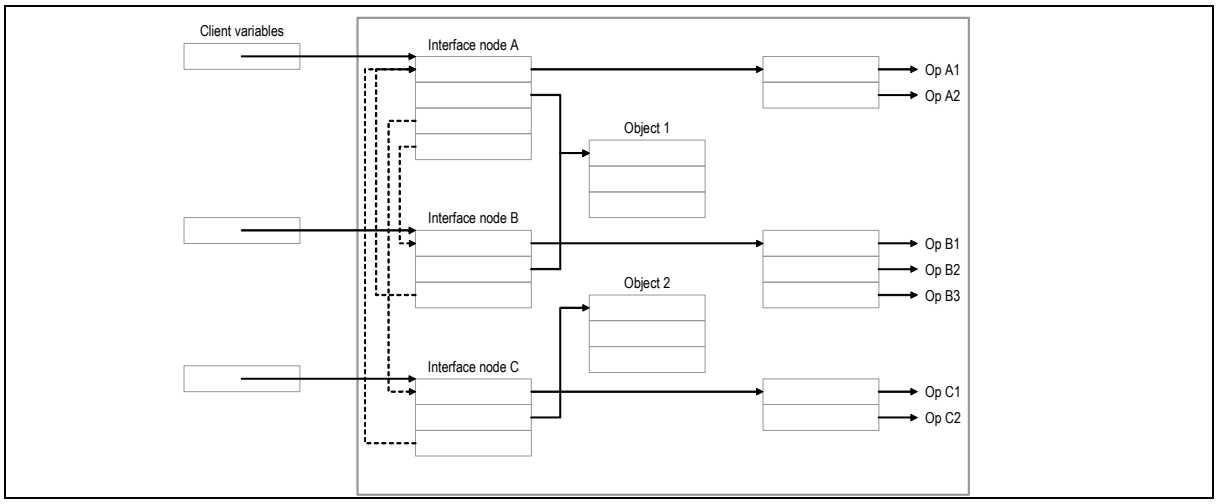

Abbildung A.3-1: ***COM-Objekt***
Quelle: *Szyperski* (2002, 332)

Wird von einem Objekt auf ein COM-Objekt eine Referenz generiert, wird der interne Referenzzähler des COM-Objektes durch die Operation AddRef inkrementiert. Sobald die Referenz wieder aufgegeben wird, wird mit der Operation Release der Referenzzähler dekrementiert. Auf diese Art wird eine kooperative Speicherbereinigung realisiert.

DCOM (Distributed COM) erweitert COM, indem die bereits bei COM gegebene Interprozesskommunikation zu einer solchen erweitert wird, welche die zwischen Rechnern vorhandene, unterschiedliche Repräsentation von Daten unterstützt. Weiterhin werden Dienste zur Gewährleistung der Sicherheit und Fehleridentifikation bei entfernten Rechnern realisiert.

COM+ erweitert die Funktionalitäten von COM mit Technologien zur Unterstützung von z.B. Transaktionen, asynchronem Nachrichtenversand und Lastverteilung.

Mit der Common Language Infrastructure[106] (CLI) wird eine sprachunabhängige Plattform zur Interoperabilität definiert. Die CLI umfasst die Spezifikation von Laufzeitdiensten z.B. zur Speicherbereinigung, das Common Type System (zur Definition eines Typsystems, CTS) sowie die Common Language Specification (CLS). Damit wird eine vollständige Abstraktion von der jeweils vorliegenden Hardware-Plattform erreicht. Die Microsoft CLR (Common Language Runtime) implementiert die Spezifikation der CLI. Im Gegensatz zur JVM unterstützt CLR neben der Abstraktion von der vorliegenden Plattform auch die Integration von

[106] Die Spezifikation der Common Language Infrastructure als Ecma International Standard ECMA-335 und ISO/IEC Standard 23271:2006 ist in der aktuellen Version auf den folgenden Seiten abrufbar (jeweils zugegriffen am 11.07.2007): http://www.ecma-international.org/publications/standards/Ecma-335.htm bzw. http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm.

Komponenten, die spezifische Eigenschaften der Hardware berücksichtigen. Somit wird in der CLR neben der COM-Interoperabilität der direkte Zugriff auf die jeweils gegebene Plattform unterstützt.

A.4 Bestimmung von Agentenklassen in Etappe 2

Teilsystem	Agentenklasse	Aufgabe	Beschreibung
User Interfaces	PersonalAssistantGuiAgent (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.9)	ReceiveDeliverDocuments	Empfangen der angeforderten Dokumente und Veranlassung ihrer Visualisierung
		lookUpEHR	Suchen der Patientenakte
		SendQueryPatientData	Initiierung der Suche nach den Dokumenten
		SendOpenEHR	Öffnen der Patientenakte
External Resources	HISWrapper (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.4)	ReceiveRequestExternalData	Empfangen einer Anfrage nach Dokumenten von einem Konnektor
		executeQuery	Ausführung der Anfrage an das Informationssystem
		transformPatientData	Transformation der Daten aus dem KIS in geeignete Ontologieobjekte
		SendDeliverDocuments	Versand der extrahierten Dokumente an den ursprünglichen Sender der Nachricht
		ReceiveQueryDocuments	Empfangen einer Anfrage nach Dokumenten von einem AMDTaskAgent
	GPISWrapper (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.3)	ReceiveQueryDocuments	Empfangen einer Anfrage nach Patientendokumenten von einem AMDTaskAgent
		executeQuery	Ausführung der Anfrage an das Informationssystem
		transformPatientData	Transformation der Daten aus dem KIS in geeignete Ontologieobjekte

External Resources	GPISWrapper	SendDeliverDocuments	Versand der extrahierten Dokumente an den ursprünglichen Sender der Nachricht
		ReceiveRequestExternalData	Empfangen einer Anfrage nach Dokumenten von einem Konnektor
Application Core	ActiveMedicalDocument (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.12)	ReceiveOpenEHR	Empfangen einer Nachricht zum Öffnen der Patientenakte
	DataWrapper (interner Management-Agent des AMD, für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.2)	ReceiveCheckAccess	Empfangen einer Nachricht zur Überprüfung des berechtigten Zugriffs
		ReceiveLocateResources	Empfangen einer Nachricht zur Lokalisierung von Ressourcen über die TI
		checkAccess	Überprüfung der Zugriffsberechtigung
		SaveTempRecord	Persistente Speicherung der aggregierten Daten aus Performanzgründen
		SendRequestLinks	Versenden einer Anfrage an den Kartenleser zur Bereitstellung von Links zu gespeicherten Patientendaten
	DataRetrievalManager (interner Management-Agent des AMD, für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.1)	SendLocateResources	Senden einer Nachricht zur Lokalisierung von Ressourcen über die TI
		SendAddNewDocuments	Senden einer Anfrage zum Hinzufügen von Dokumenten
		ReceiveDeliverDocuments	Empfangen von angefragten Dokumenten
		GenerateTaskAgent	Erzeugen von Agenten für die Datenaggregation
		SendRequestPatientData	Senden einer Anfrage an den AMDTaskAgent

Application Core	AMDTaskAgent (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.11)	ReceiveRequestPatientData	Empfangen einer Anfrage zur Aggregation von Daten
		SendRequestPatientData	Bearbeitung einer Anfrage zur entfernten Extraktion
		SendQueryDocuments	Bearbeitung einer Anfrage zur lokalen Extraktion
		ReceiveDeliverDocuments	Empfangen und Weiterverarbeitung der extrahierten Daten
		SendDeliverDocuments	Weiterleitung der empfangenen Daten
	ViewManager (interner Agent des AMD, für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.5)	ReceiveDeliverDocuments	Empfangen von Daten zur Weiterverarbeitung
		ProcessDataToGUI	Aufbereitung der Daten entsprechend den Anforderungen des Anzeigegerätes
		SendDeliverDocuments	Weiterleitung der empfangenen und aufbereiteten Dokumente an den ursprünglichen Initiator der Anfrage
	GPConnectorAgent (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.6)	RequestLinkBehaviour	Weiterleitung der Anfrage zum Lesen der auf der aktuell eingelegten eGK gespeicherten Verweisliste an den Kartenleser
		SendReceiveExternal-RequestBehaviour	Weiterleitung der Anfrage nach den Dokumenten an einen entfernten Konnektor
		ReceiveSendExternal-RequestBehaviour	Empfangen einer Anfrage von einem externen System zur Aggregation von Daten

Application Core	HConnectorAgent (für eine Beschreibung der Verarbeitungsvorschrift siehe Anhang A.5.7)	ReceiveSendExternal-RequestBehaviour	Empfangen einer Anfrage von einem externen System zur Aggregation von Daten
		RequestLinkBehaviour	Weiterleitung der Anfrage zum Lesen der auf der aktuell eingelegten eGK gespeicherten Verweisliste an den Kartenleser
		CheckAccessBehaviour	Überprüfen der Anfrage, ob das anfragende externe Informationssystem Zugriffsberechtigung besitzt
		CardReaderStatusRequest-Behaviour	Empfangen einer Anfrage zur Bereitstellung der Informationen über die aktuell im Lesegerät eingelegten Karten
		SendReceiveExternal-RequestBehaviour	Weiterleitung der Anfrage nach den Dokumenten an einen entfernten Konnektor

Tabelle A.4-1: ***Bestimmung von Agentenklassen***
Quelle: Eigene Darstellung, in Anlehnung *Bastian* (2005, 92-94)

A.5 Detaillierte Beschreibung elementarer Agenten

Die in diesem Abschnitt dargestellten Beschreibungen für ausgewählte zentrale Agenten sind an die Notation bei *Reinke* (2003, C-1-C-8) angelehnt. Dabei werden einerseits Eigenschaften mit textuellen Mitteln beschrieben. Andererseits wird die wesentliche Eigenschaft des Agentenverhaltens durch Aktivitätsdiagramme repräsentiert. In diesen Diagrammen werden Behaviours mit Interaktionen durch einen grau hinterlegten, Methoden und Behaviours ohne Interaktionen durch einen weiß hinterlegten Aktionszustand dargestellt. Die Sichtbarkeit und Typen von Attributen und Operationen werden in Java-Notation angegeben. Bei der Signatur von Operationen wird aus Übersichtlichkeitsgründen auf die Angabe der Parameterliste verzichtet.

A.5.1 DataRetrievalManager

Component DataRetrievalManager	
Description	The DataRetrievalManager is the central element for the coordination of several task agents dealing with the retrieval of patients' documents. Having received a request for the aggregation of documents these requests are stored in a temporary list. Before the documents can be retrieved, their location is made available via the DataWrapper agent. For each document to be received, a new task agent is created. These task agents are responsible for actually retrieving the documents from the distributed information systems. Once these documents are received from the various task agents, they are forwarded by the DataRetrievalManager to the DataWrapper agent.
Relations	**inherit** AMDManagementAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = ActionTask, InteractionTask (for differentiation see Abbildung A.5-1); Ontology = ASAinlog; Locality = stationary;

Attributes	// Stores a list of types of REQUESTs for documents to be processed, sorted by the requesting AIDs
	private Hashtable actualQueryDocsRequests
	// Stores a list of requesting AIDs, sorted by addresses
	private Hashtable nameToAddressMapping

Operations	**private void** sendErrorMsg **protected void** setup **protected void** takeDown

Tabelle A.5-1: ***Darstellung der DataRetrievalManager-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

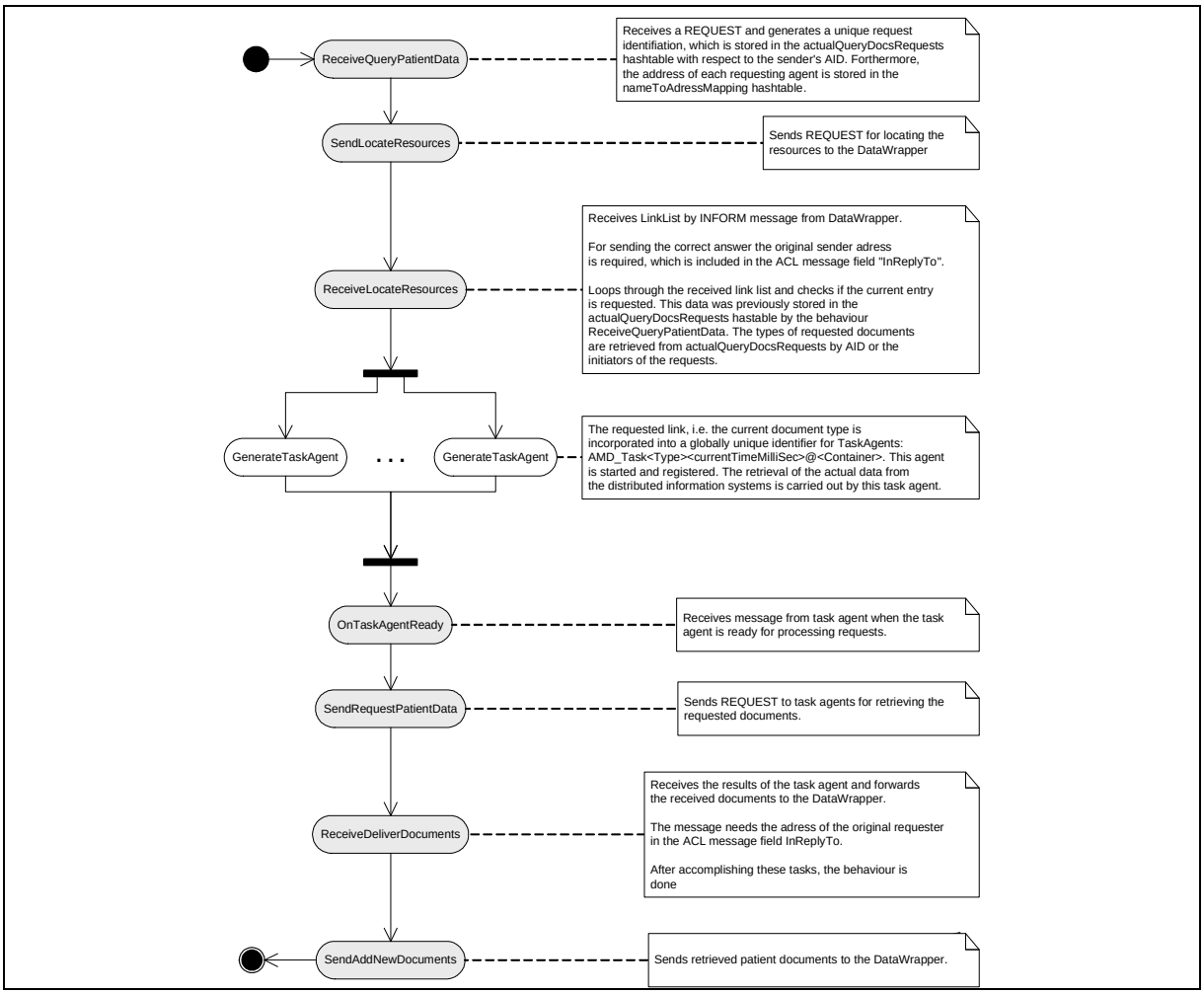

Abbildung A.5-1: Darstellung der DataRetrievalManager-Aktivität
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.2 DataWrapper

Component DataWrapper	
Description	The DataWrapper is the single point of access to data stored within an active medical document. If patient data is needed, the request is forwarded to the DataWrapper. The pointers to the patients' documents are retrieved from the electronic patient card via the connector component and are sent to the DataRetrievalManager for further processing. The received documents are sent to the ViewManager.
Relations	**inherit** AMDManagementAgent

AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = ActionTask, InteractionTask (for differentiation see Abbildung A.5-3); Ontology = ASAinlog; Locality = stationary;

Attributes	// Ontology object containing the complete electronic patient record.
	private MedicalInformation electronicPatientRecord;
	// List of agents, which are granted access to the documents of the patient. Currently, these are the agents
	// DataWrapper, DataRetrievalManager, ViewManager, and AppointmentManager.
	protected List allowedAgents;

Operations	// Get a reference to the connector of the hospital.
	private AID lookupConnector;
	private void sendErrorMsg;
	private synchronized void storeInPatientRecord;
	// Check whether the given agent is granted access to the DataWrapper
	protected boolean checkAccess;
	// Close the DataWrapper
	public void takeItDown

Tabelle A.5-2: ***Darstellung der DataWrapper-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

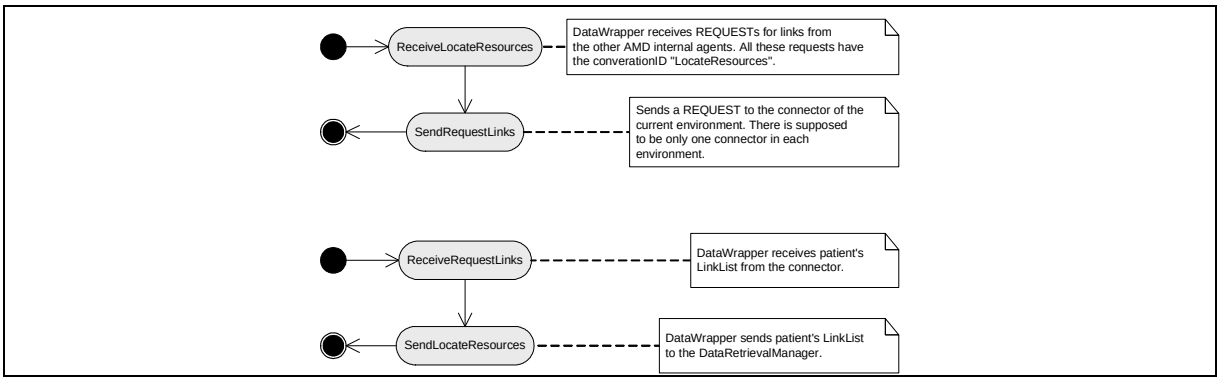

Abbildung A.5-2: ***Darstellung der DataWrapper-Aktivität (1)***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

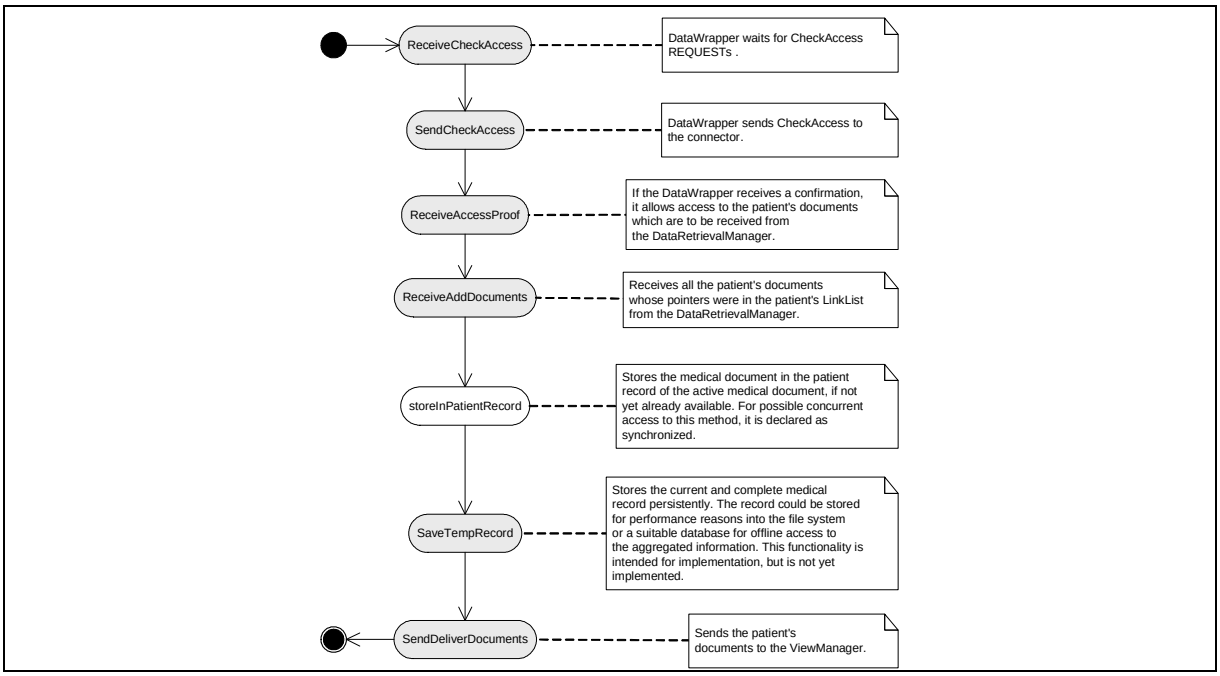

Abbildung A.5-3: Darstellung der DataWrapper-Aktivität (2)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.3 GPISWrapper

Component GPISWrapper		
Description	The GPISWrapper receives requests for extracting medical documents. These requests are either issued by task agents or a connector. According to this differentiation, two different behaviours are implemented dealing with requests from task agents or a connector. After receiving the request, the retrieval of data out of a general practitioner's information system is simulated by reading patient documents from the file system. These data are transformed from an HL7 CDA data structure to an adequate ontology object. This ontology object is finally sent back to the original sender of the request. Hence, the wrapper agent implements the transformation of an agent request into a proprietary system call. The result of this call is transformed into an agent message.	
Relations	**inherit** AMDWrapperAgent	
AgentProfile	Lifecycle =	AgentLifeCycle;
	TaskTypes =	WrapperTask, InteractionTask (for differentiation see Abbildung A.5-4 and Abbildung A.5-5);
	Ontology =	ASAinlog;
	Locality =	stationary;

Attributes	// Inherited attributes **private** ReceiveQueryDocuments _ReceiveQueryDocuments; **private** ReceiveRequestExternalData _ReceiveRequestExternalData; **protected** Codec codec; **public** String patientID; **public** String environment; **public** SendDeliverDocuments _SendDeliverDocuments;
Operations	**public void** setup **public void** executeQuery // Inherited operations **public** takeDown **public** MedicalDocument transformPatientData **public** String openPatientDataFromXMLFile **public void** sendErrorMsg **public** Codec getCodec

Tabelle A.5-3: ***Darstellung der GPISWrapper-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

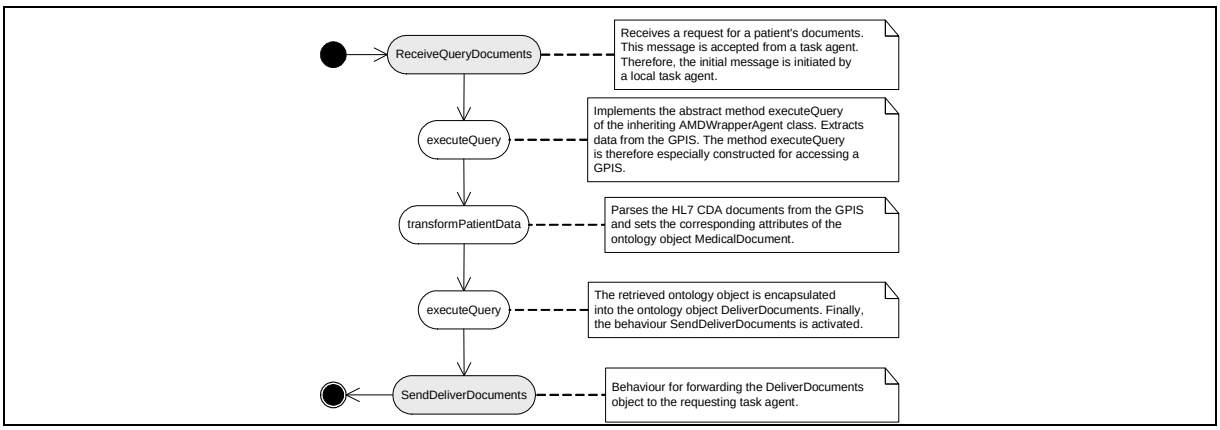

Abbildung A.5-4: ***Darstellung der GPISWrapper-Aktivität (1)***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

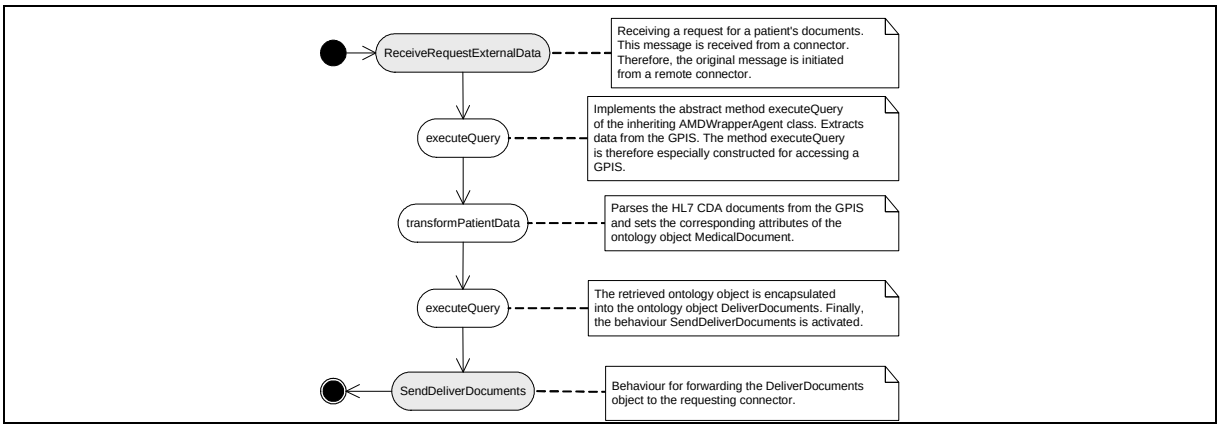

Abbildung A.5-5: Darstellung der GPISWrapper-Aktivität (2)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.4 HISWrapper

Component HISWrapper		
Description	The HISWrapper receives requests for extracting medical documents from the wrapped SAP IS-H resp. i.s.h.med information system. These requests are either issued by task agents or a connector. According to this differentiation, two different behaviours are implemented dealing with requests from task agents or a connector. After receiving the request, the retrieval of data out of the chosen hospital information system is partly implemented with fully fledged functionality. Other parts of reading data from the hospital information system are only simulated, since necessary BAPIs are not available for accessing data of the deployed information system. Therefore, some of the data is read from the local file system. Data are transformed from an HL7 CDA data structure to an adequate ontology object. This ontology object is finally sent back to the original sender of the request. Hence, the wrapper agent implements the transformation of an agent request into a proprietary system call. The result of this call is transformed into an agent message. Furthermore, the HISWrapper is capable of processing requests for retrieving patients' and resources' schedules. Appointments are read from and written to a hospital information system by simulated BAPIs, since necessary BAPI functionality is not implemented in the deployed information system. Additionally, the changed patient's health state can be retrieved from and written to the (simulated) hospital information system.	
Relations	**inherit** AMDWrapperAgent	
AgentProfile	Lifecycle =	AgentLifeCycle;
	TaskTypes =	WrapperTask, InteractionTask (for differentiation see Abbildung A.5-6, Abbildung A.5-7, Abbildung A.5-8, and Abbildung A.5-9);
	Ontology =	ASAinlog;
	Locality =	stationary;

Attributes	**private** SAPJCoConnection SAPobj; // Inherited **private** ReceiveQueryDocuments _ReceiveQueryDocuments; **private** ReceiveRequestExternalData _ReceiveRequestExternalData; **protected** Codec codec; **public** String patientID; **public** String environment; **public** SendDeliverDocuments _SendDeliverDocuments;
Operations	**private** List getAllDiagnosisAsList **private** List getAllFindindsAsList **private** String getInternalSAPiD **private** String transformSAPXML2CDArev2 **private** List getAllCaseids **public void** setup **public void** executeQuery **public void** executeQueryScheduleForPatient **public void** executeQueryHealthState **public void** executeQueryScheduleForResource **public** DeliverInitState getHealthState **public void** writeBackChangedSchedule **public** DeliverSchedule getAllResourceAppointments **public** List getAllAppointments // Inherited operations **public** takeDown **public** MedicalDocument transformPatientData **public** String openPatientDataFromXMLFile **public void** sendErrorMsg **public** Codec getCodec

Tabelle A.5-4: ***Darstellung der HISWrapper-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

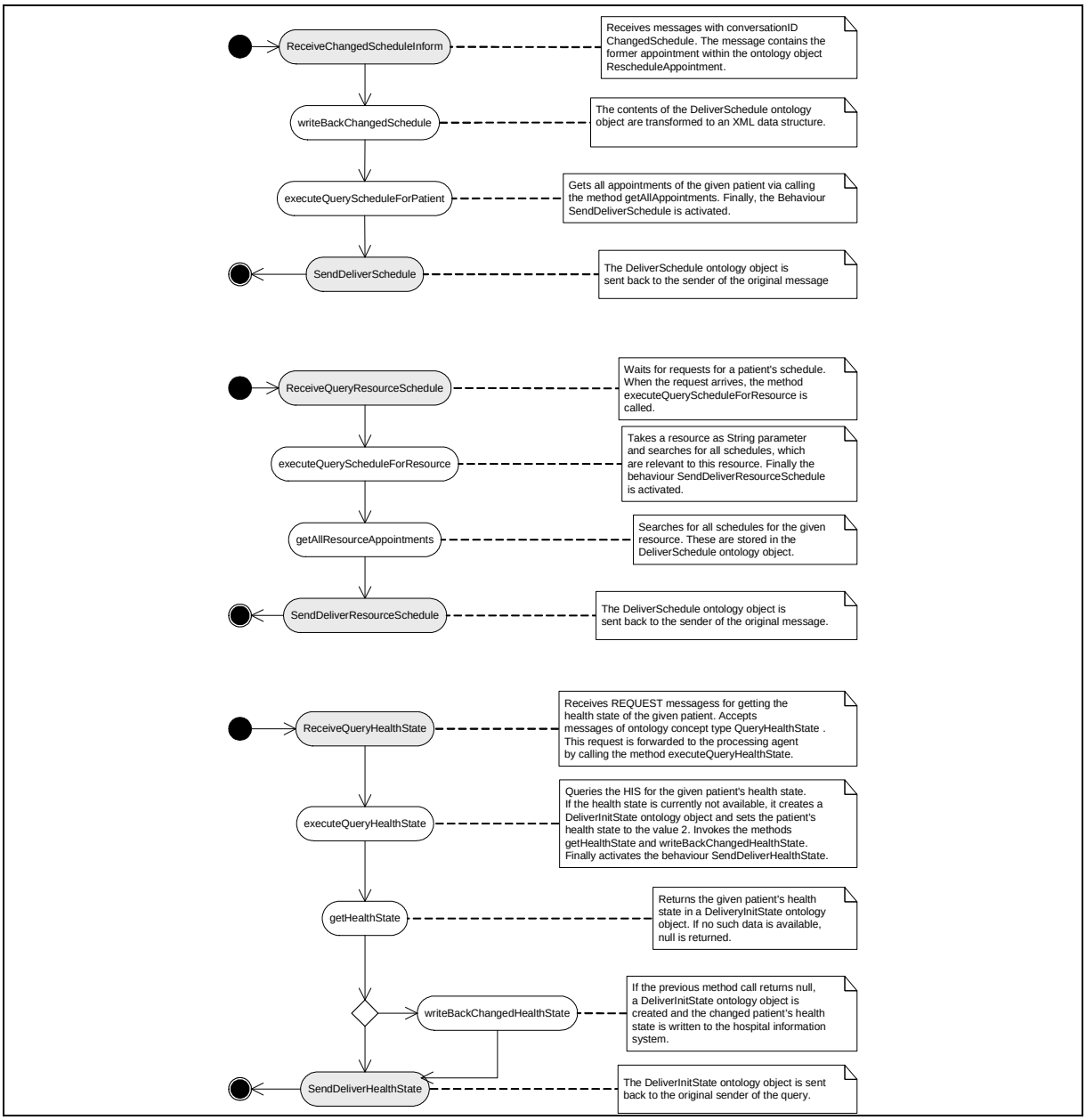

Abbildung A.5-6: Darstellung der HISWrapper-Aktivität (1)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

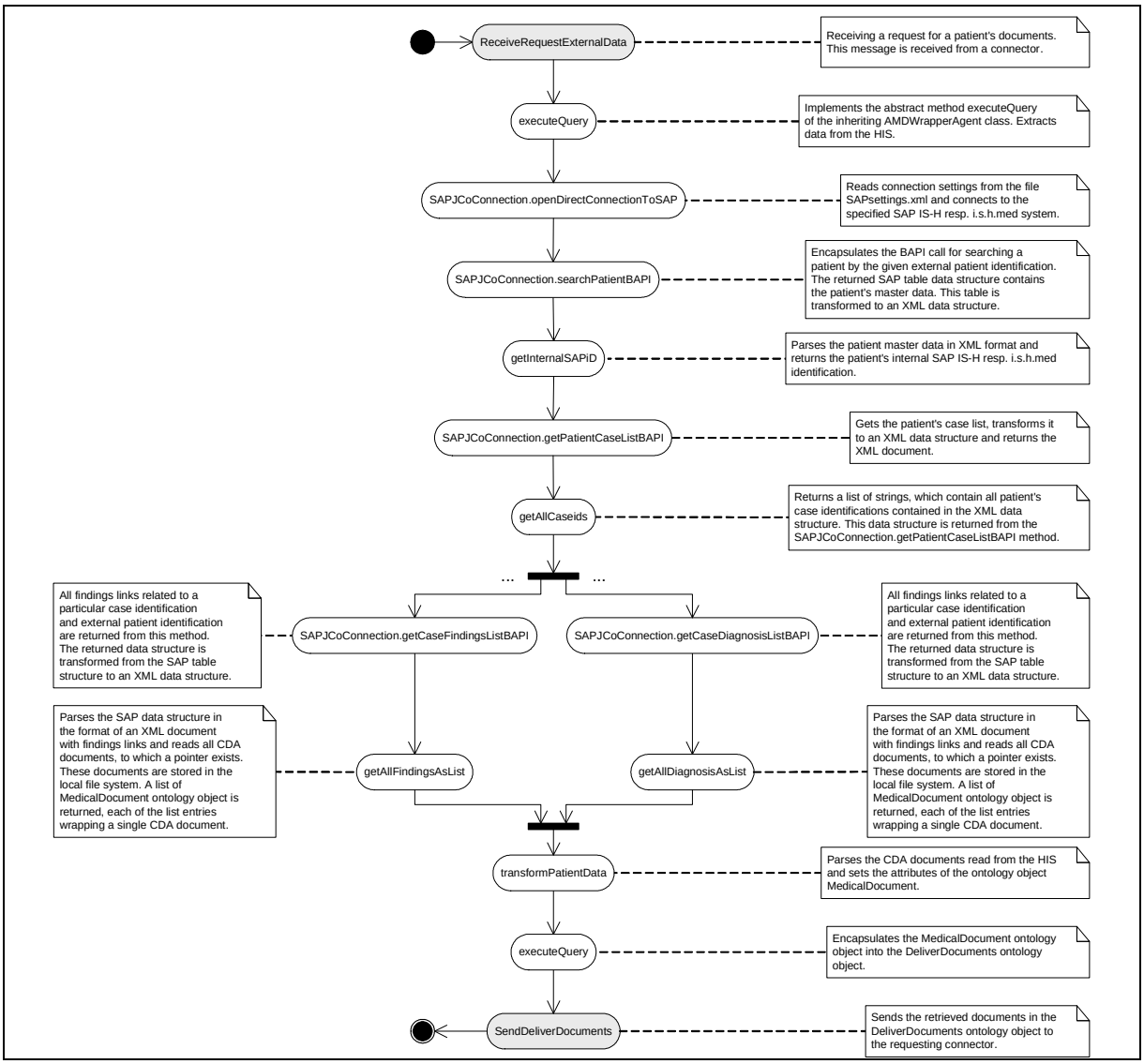

Abbildung A.5-7: Darstellung der HISWrapper-Aktivität (2)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

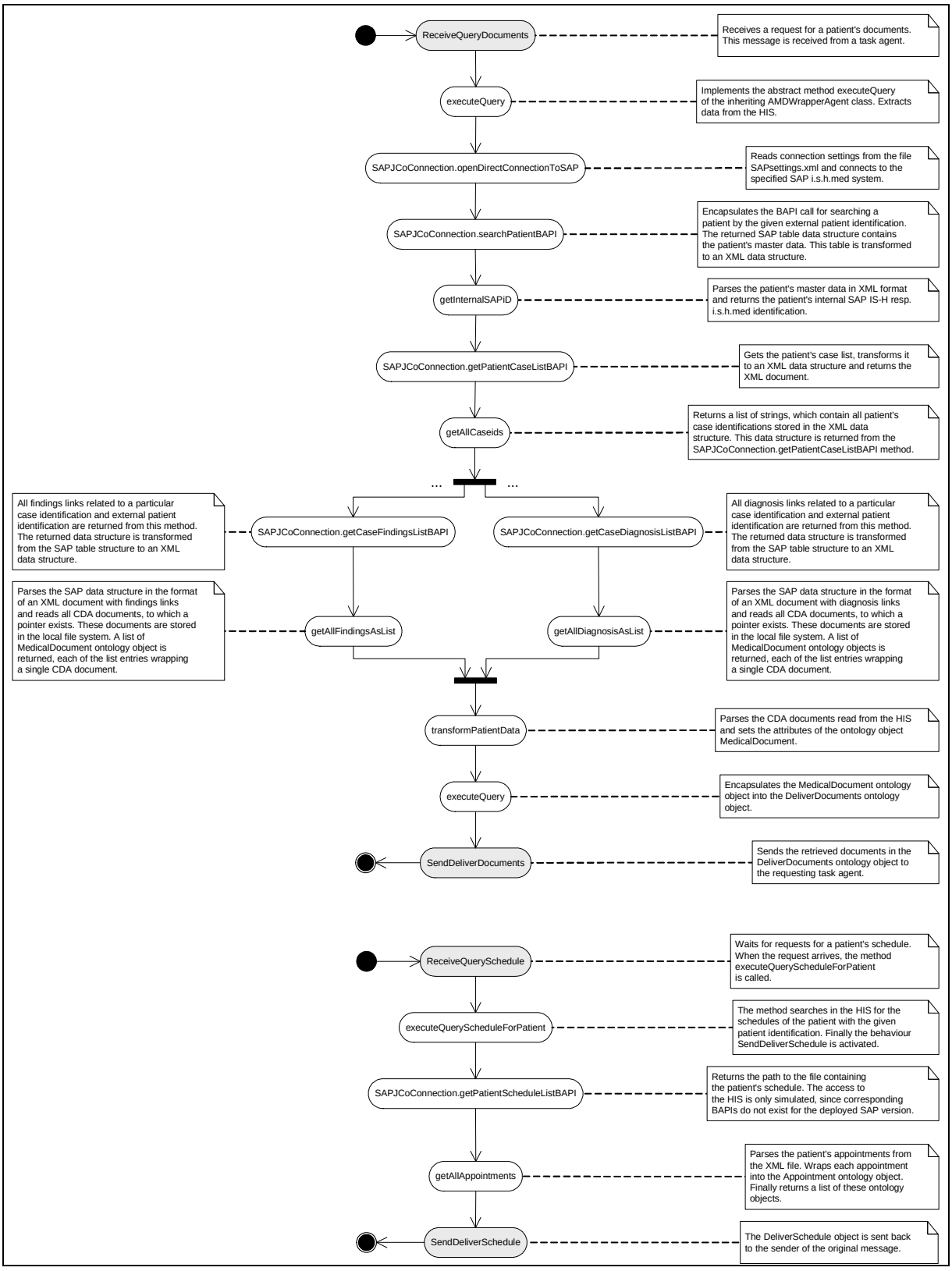

Abbildung A.5-8: Darstellung der HISWrapper-Aktivität (3)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

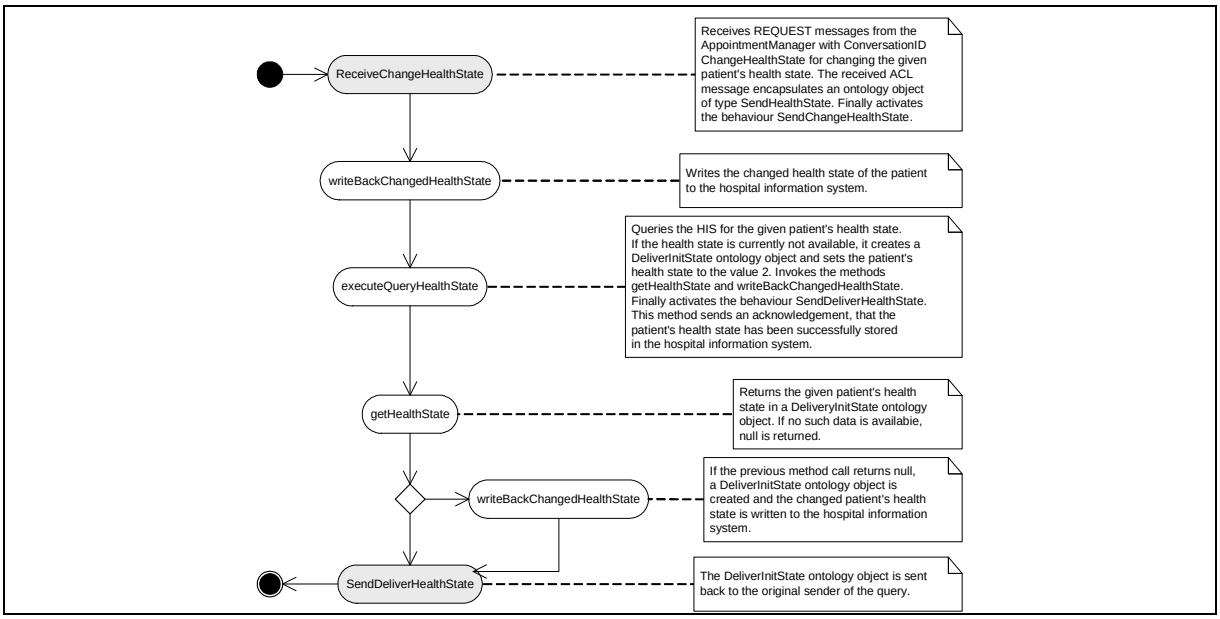

Abbildung A.5-9: Darstellung der HISWrapper-Aktivität (4)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.5 ViewManager

Component ViewManager	
Description	The ViewManager receives documents from the DataWrapper agent. These documents are to be transformed according to the display capabilities of the user interface of the currently used device. Up to now, this functionality is not yet implemented, but already intended for implementation. In the current implementation, the received documents are finally sent to the initiator of the request. Furthermore, the agent receives updated schedules from the AppointmentManager and forwards these to the initial requester by activating a special behaviour.
Relations	**inherit** AMDManagementAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = ManagementTask, InteractionTask (for differentiation see Abbildung A.5-10); Ontology = ASAinlog; Locality = stationary;
Attributes	None

Operations	// Inherited operations **private void** sendErrorMsg **protected void** setup **protected void** takeDown

Tabelle A.5-5: ***Darstellung der ViewManager-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

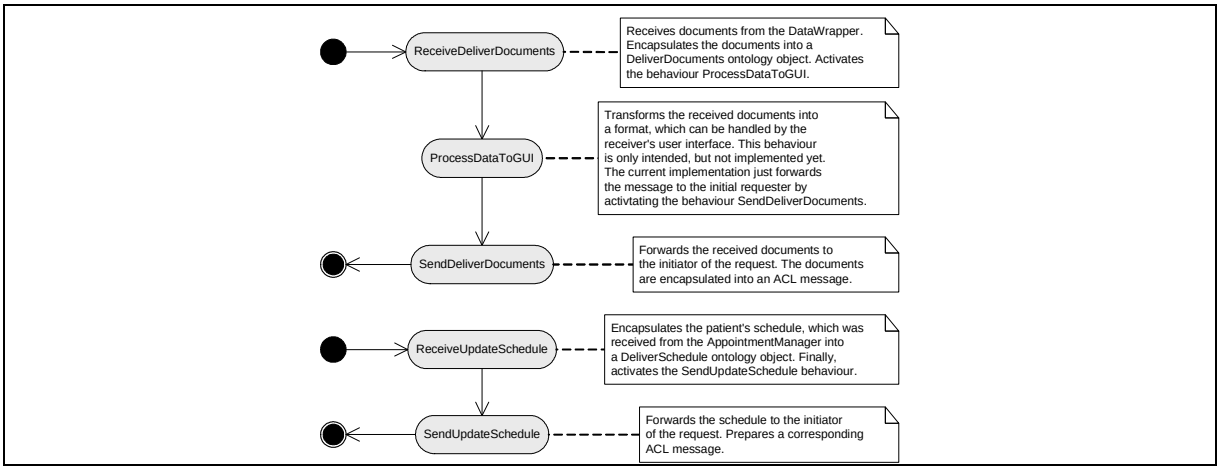

Abbildung A.5-10: ***Darstellung der ViewManager-Aktivität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.6 GPConnectorAgent

Component GPConnectorAgent	
Description	Receives requests for obtaining the link list. Furthermore, sends requests to external connectors for data retrieval of medical documents. Finally, receives requests from external information systems for data retrieval.
Relations	**inherit** Agent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask; Ontology = ASAinlog; Locality = stationary;

Attributes	// Reference to the SecurityHelper providing security mechanisms
	private SecurityHelper mySecurityHelper;
	// Denotes, whether the telematics infrastructure can currently be accessed.
	private boolean tm = **false**;
	public static final String GET_PRINCIPAL;

Operations	// Inherited operations **protected void** setup **protected void** takeDown

Tabelle A.5-6: ***Darstellung der GPConnectorAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

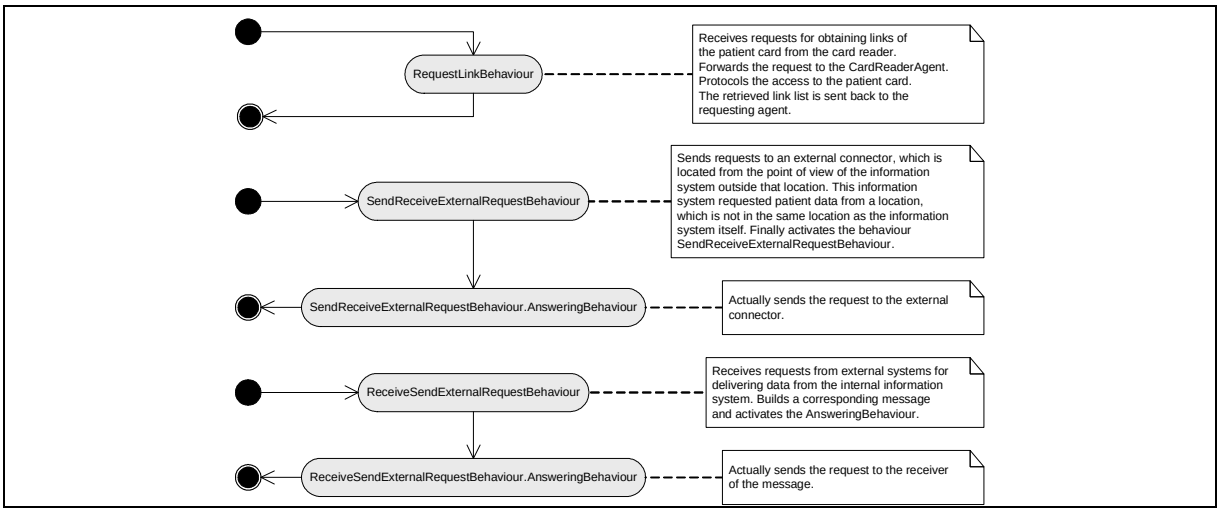

Abbildung A.5-11: ***Darstellung der GPConnectorAgent-Aktivität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.7 HConnectorAgent

Component HConnectorAgent	
Description	Receives requests for patient documents from an external system. Furthermore, receives requests for accessing the link list of the patient card. Additionally, granting the access to the patient card can be checked. In order to get to know, which cards are currently inserted into the card reader, its status can be requested. Finally, the connector can send a request to external information systems for retrieving the patient documents.
Relations	**inherit** Agent

AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask; Ontology = ASAinlog; Locality = stationary;

Attributes	// Reference to the SecurityHelper providing security mechanisms
	private SecurityHelper mySecurityHelper;
	// Denotes, whether the telematics infrastructure can currently be accessed.
	private boolean tm = **false**;
	public static final String GET_PRINCIPAL;

Operations	// Inherited operations **protected void** setup **protected void** takeDown

Tabelle A.5-7: ***Darstellung der HConnectorAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

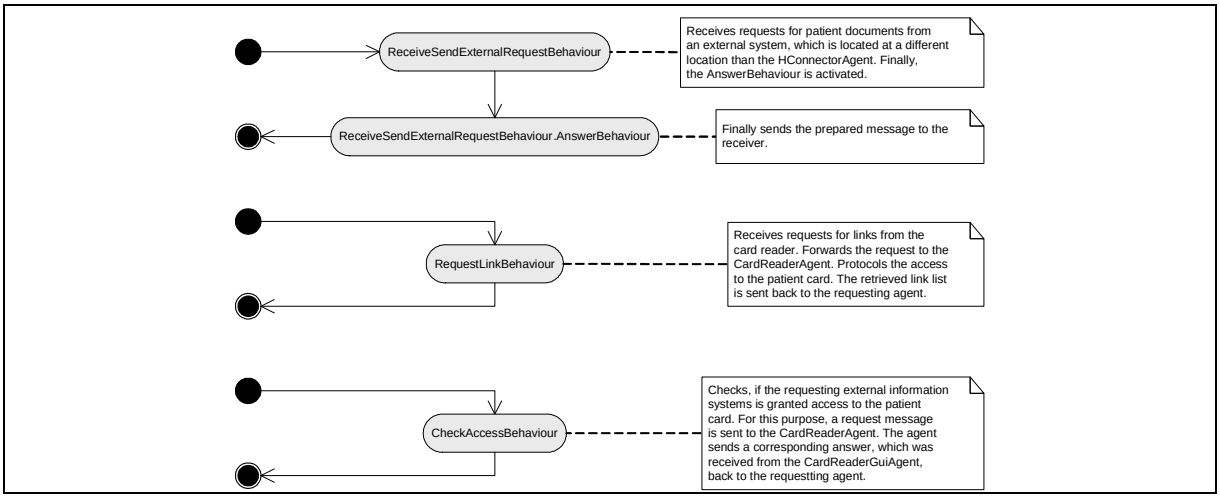

Abbildung A.5-12: ***Darstellung der HConnectorAgent-Aktivität (1)***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

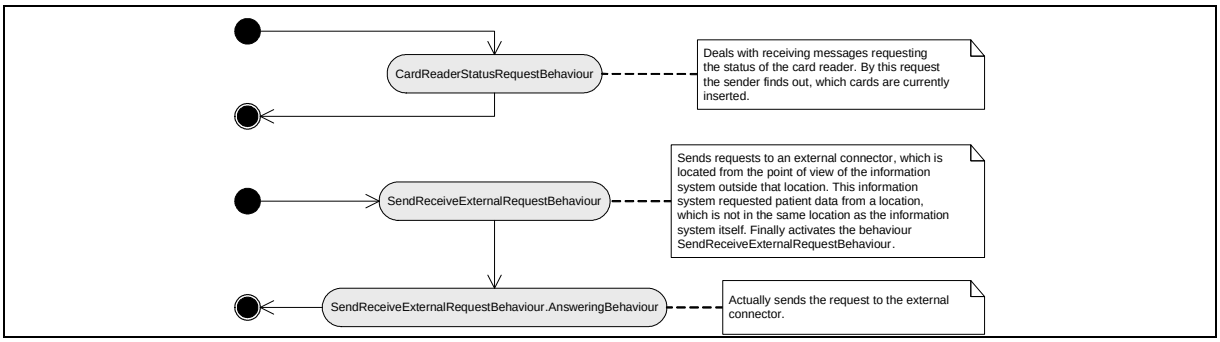

Abbildung A.5-13: Darstellung der HConnectorAgent-Aktivität (2)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.8 CardReaderGuiAgent

Component CardReaderGuiAgent	
Description	The CardReaderGuiAgent deals with three different requests in a parallel fashion. The behaviours are activated throughout the complete lifecycle of the agent. Therefore, a cyclic behaviour is implemented for each of the functionalities. The agent is capable of receiving requests reading the link list from the patient card. Furthermore, the data from the currently inserted patient and user cards can be requested. In order to allow for the correct functionality of the card reader, the requested patient and user data are compared to the currently inserted patient and user card. Only if these data are equivalent, successful access to the card reader and its cards is granted.
Relations	**inherit** GuiAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask; Ontology = ASAinlog; Locality = stationary;

Attributes	**private** SecurityHelper mySecurityHelper;
	// Ontology object representing the current status of the card reader.
	private CardReaderStatus cardReaderStatus;
	// Denotes the IP address of the host.
	private static InetAddress thisHost;
	// The host name
	private static String hostname;

Attributes	// A reference to the user who currently uses the GUI
	protected AMDPrincipal userPrincipal;
	// A reference to the GUI of the (simulated) card reader
	public JPersonalAssistantGuiCardReader gui;

Operations	**public void** setup **protected void** takeDown **public void** startNewGUI **protected void** onGuiEvent **public** CardReaderStatus getCardReaderStatus

Tabelle A.5-8: ***Darstellung der CardReaderGuiAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

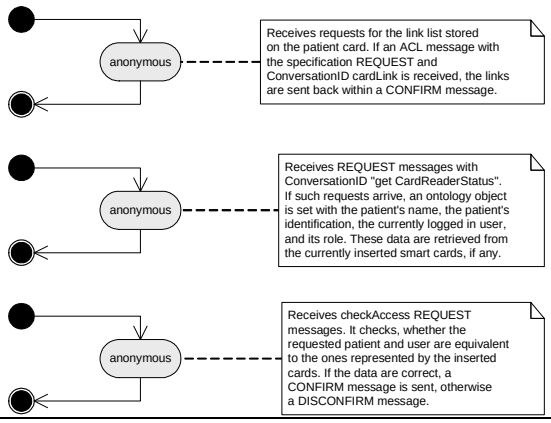

Abbildung A.5-14: ***Darstellung der CardReaderGuiAgent-Aktivität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.9 PersonalAssistantGuiAgent

Component PersonalAssistantGuiAgent	
Description	Initiates the search for the patient's documents and the associated schedule. Receives documents and schedules to be displayed at the user interface. Forwards messages for rescheduling appointments. Forwards messages to change, delete, or create new appointments and to change the patient's health status. Receives messages for displaying patient's or resource's schedules and calls corresponding controller methods. Enables push and pull functionality for mobile devices for aggregating the patient data.
Relations	**inherit** AMDPersonalAssistantAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask; UserInterface; ActionTask (for a differentiation see Abbildung A.5-15, Abbildung A.5-16, Abbildung A.5-17, Abbildung A.5-18, and Abbildung A.5-19); Ontology = ASAinlog; Locality = stationary;
Attributes	**private** JPersonalAssistantGui gui; **private static** InetAddress thisHost; **private boolean** killedAMD; **private** ActiveMedicalDocument amd; AID env; **public** ReceiveDeliverDocuments recDeliverDocs; **public static** String hostName; // Inherited **private** SendQueryPatientData _SendQueryPatientData **protected** AID[] foundQueryDocAgents; **protected** AID[] foundSchedulingAgents; **public** SendOpenEHR _SendOpenEHR; **public** String patientID; **public** Codec codec;
Operations	**private void** lookUpStartResourceSchedulingTaskAgent **private void** startAMDInstance **public void** startAMD **public void** killAMD **public void** sendSearchCallToMobileApp

Operations	**public void** sendQueryPatientData
	public void sendQueryResourceSchedule
	public void sendReschedule
	public void sendQueryPatientSchedule
	public void sendCreateNewAppointment
	public void sendChangeAppointment
	public void sendDeleteAppointment
	public Object getPatient
	public GuiRepository getResources
	public void sendChangeHealthState
	// Inherited
	public String lookUpEHR
	public String lookUpEHRSchedule
	public AID[] getFoundEHRs
	public void takeDown

Tabelle A.5-9: ***Darstellung der PersonalAssistantGuiAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

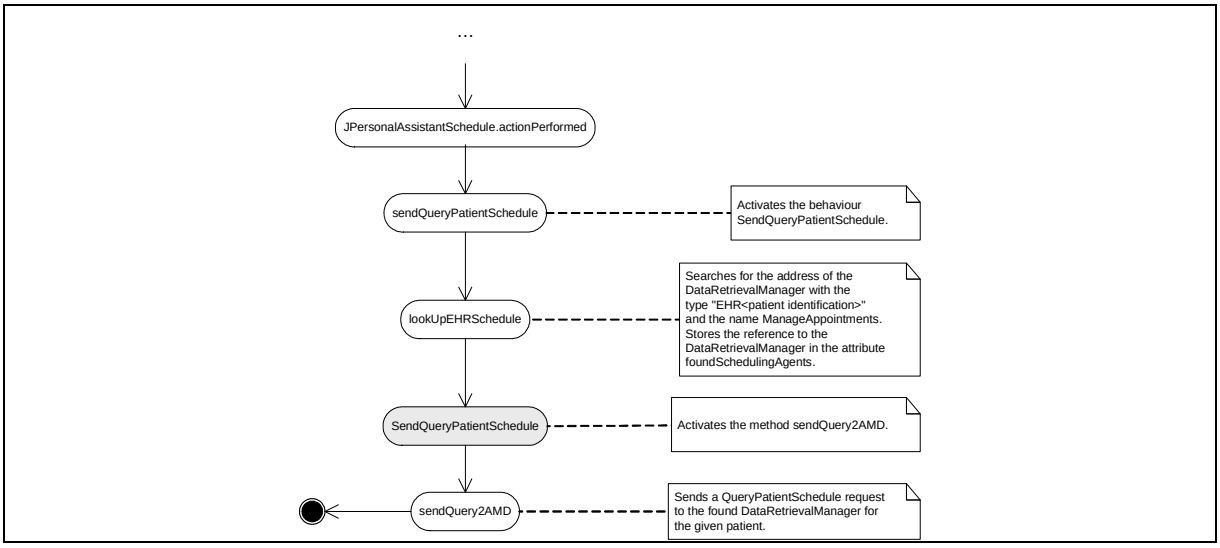

Abbildung A.5-15: ***Darstellung der PersonalAssistantGuiAgent-Aktivität (1)***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

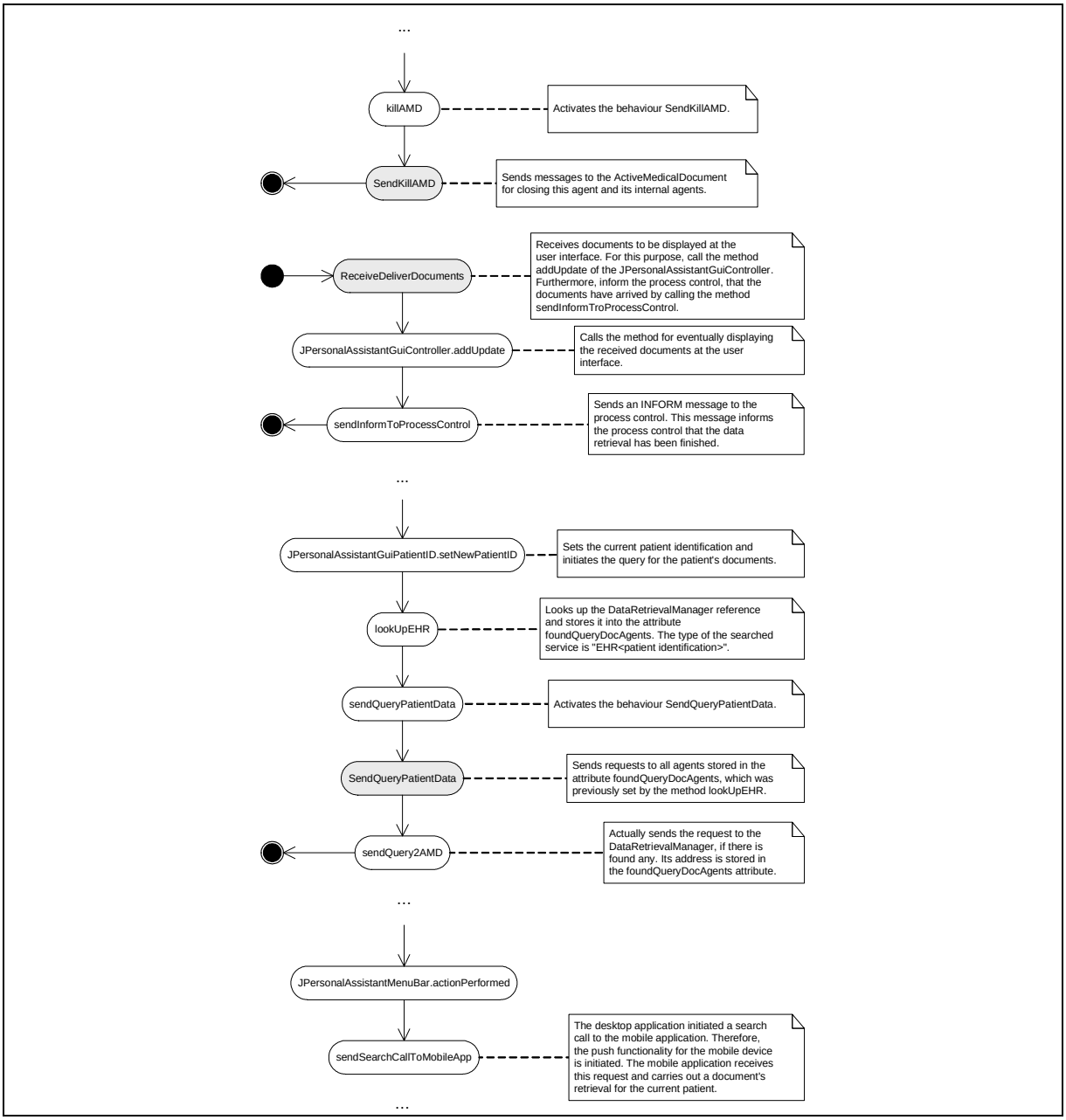

Abbildung A.5-16: Darstellung der PersonalAssistantGuiAgent-Aktivität (2)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

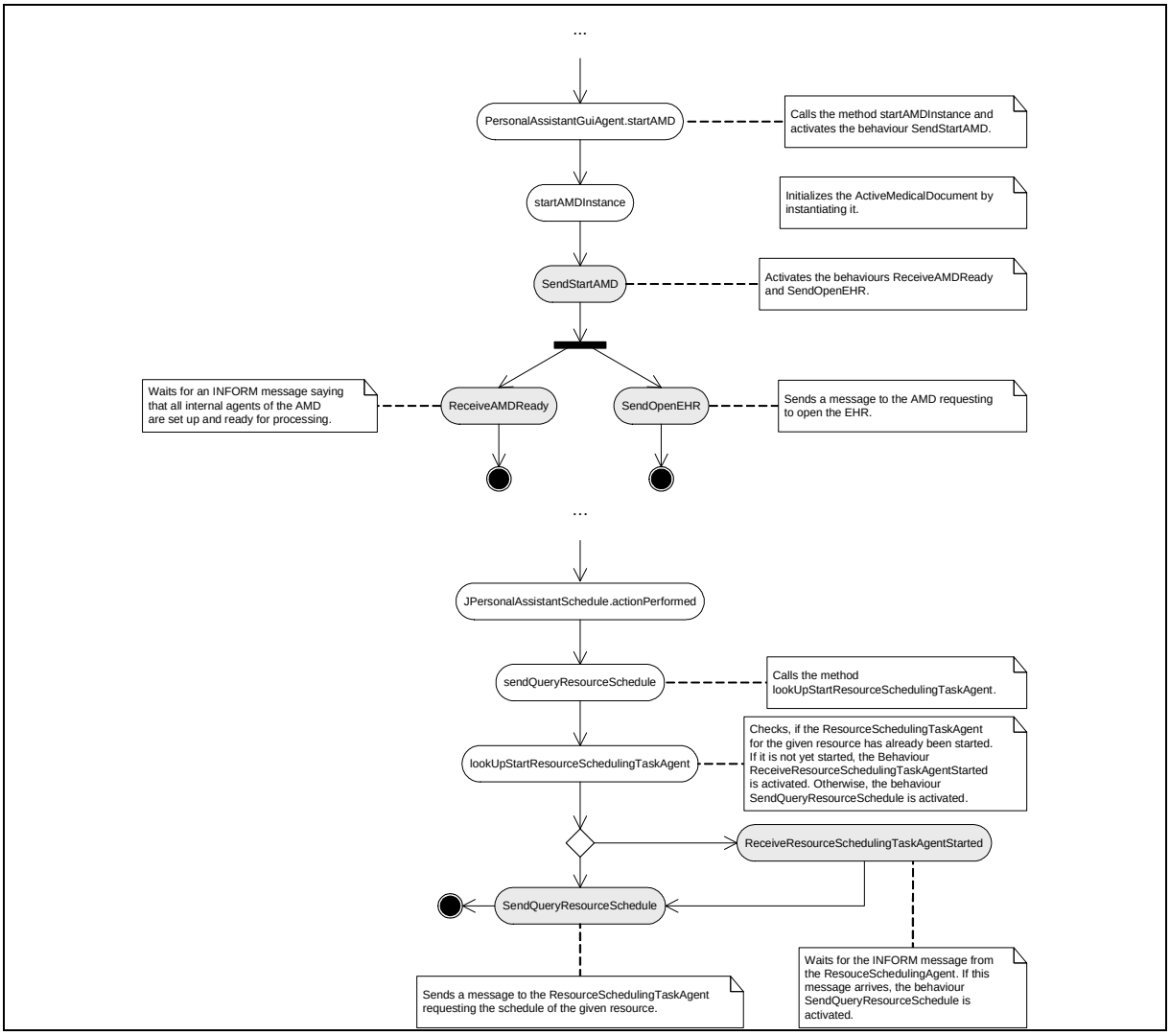

Abbildung A.5-17: Darstellung der PersonalAssistantGuiAgent-Aktivität (3)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

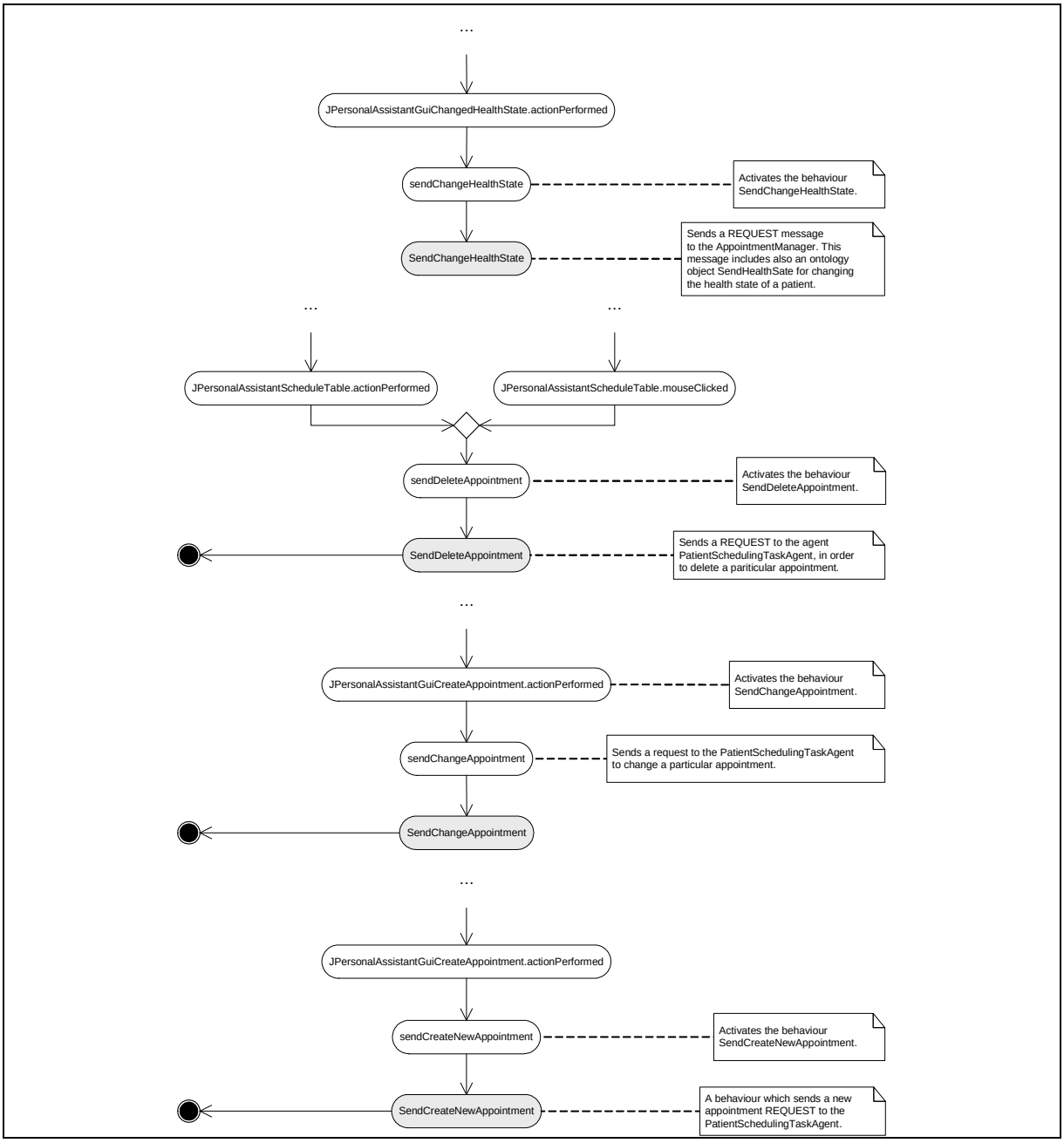

Abbildung A.5-18: Darstellung der PersonalAssistantGuiAgent-Aktivität (4)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

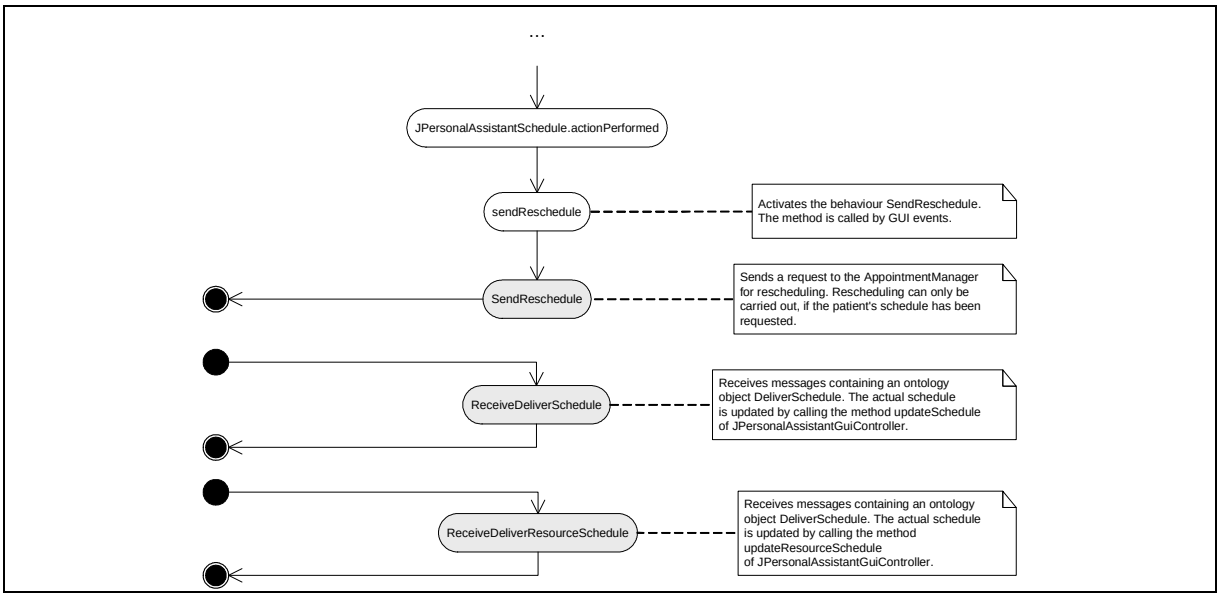

Abbildung A.5-19: Darstellung der PersonalAssistantGuiAgent-Aktivität (5)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.10 PersonalAssistantGuiAgent (mobil)

Component PersonalAssistantGuiAgent (mobil)	
Description	Implements the mobile counterpart of the PersonalAssistantGuiAgent. After a successful login to the mobile application, the behaviour for retrieving the patient's documents is activated. This behaviour waits for push requests from the desktop application. Forwards the request for getting the data of the currently inserted user and patient cards. Processes the request to retrieve the patient's documents by forwarding the request to the desktop application (pull functionality). Displays the received documents at the mobile device.
Relations	**inherit** AMDPersonalAssistantAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask; UserInterface; ActionTask (for a differentiation see Abbildung A.5-20); Ontology = ASAinlog; Locality = stationary;
Attributes	**private** PersonalAssistantGui gui; **private static** String hostName; Behaviour ExternalCallListener; Behaviour sendQueryPatientDataBehaviour; // Inherited **protected** AID[] foundQueryDocAgents; **public** String patientID;
Operations	**protected void** startExternalCallListener **protected void** removeExternalCallListener **protected void** setup **public void** getCardReaderStatus **public void** sendQueryPatientData // Inherited **public** String lookUpEHR **public** takeDown

Tabelle A.5-10: ***Darstellung der PersonalAssistantGuiAgent-Funktionalität (mobil)***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

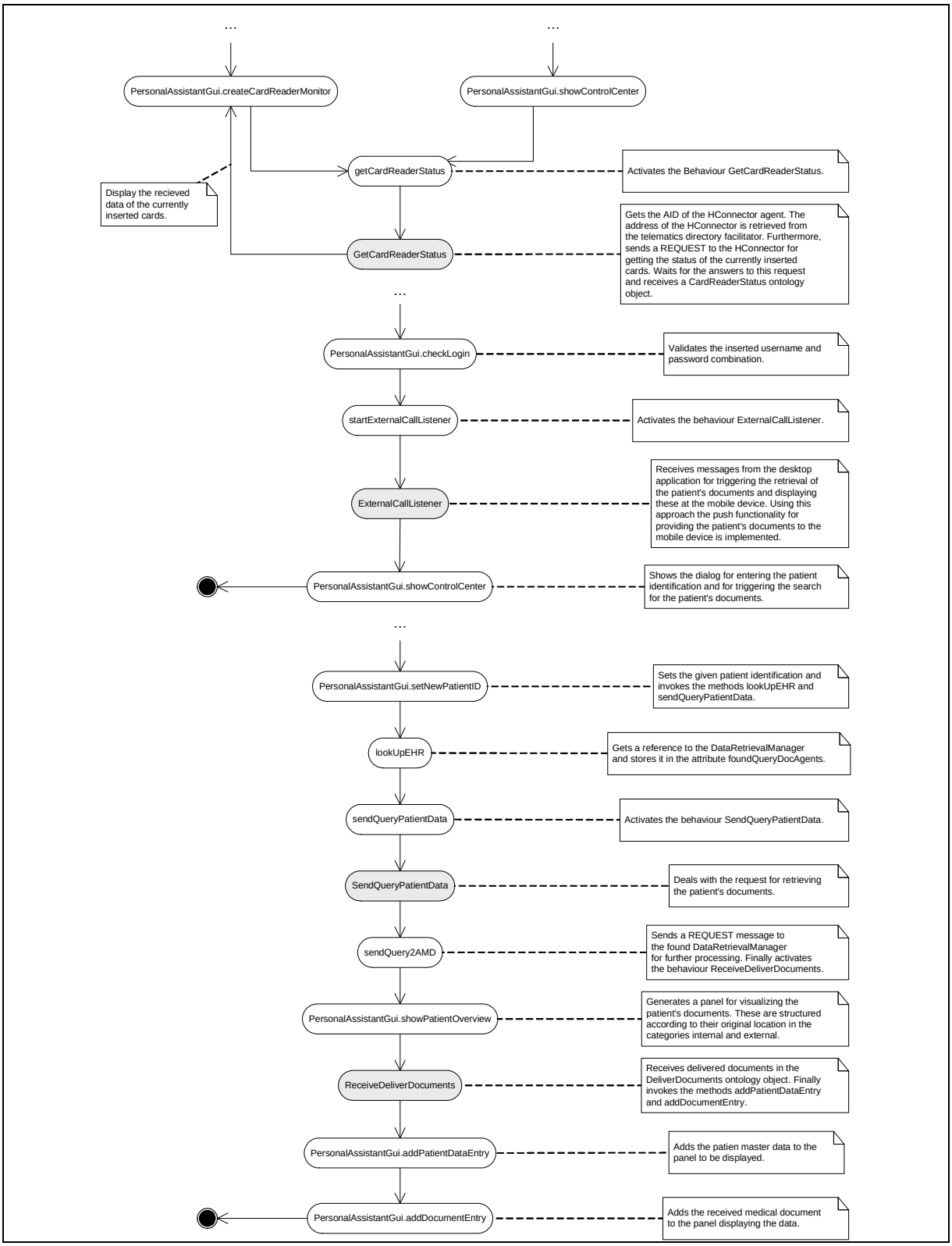

Abbildung A.5-20: Darstellung der PersonalAssistantGuiAgent-Aktivität (mobil)
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.11AMDTaskAgent

Component AMDTaskAgent	
Description	Receives requests from the DataRetrievalManager for aggregating patient data. Forwards the requests to a wrapper agent or connector agent. If data is to be retrieved from a local location, the behaviour SendQuery-Documents is activated. If data is requested from a remote location, the behaviour SendRequestPatientData is activated. Receives the actually retrieved data from a wrapper agent or a connector agent and forwards these documents to the DataRetrievalManager.
Relations	**inherit** Agent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = ActionTask; InteractionTask (for a differentiation see Abbildung A.5-21); Ontology = ASAinlog; Locality = stationary;
Attributes	**private** AID environment; **private** String actClassName; **private** ReceiveRequestPatientData _ReceiveRequestPatientData; **private** ReceiveDeliverDocuments _ReceiveDeliverDocuments;
Attributes	**public** String patientID; **public** SLCodec codec; **public** EvaluateResult _EvaluateResult;
Operations	**private** AID lookupConnector **public** String getPatientID **public** SLCodec getCodec **private void** sendErrorMsg

Tabelle A.5-11: ***Darstellung der AMDTaskAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

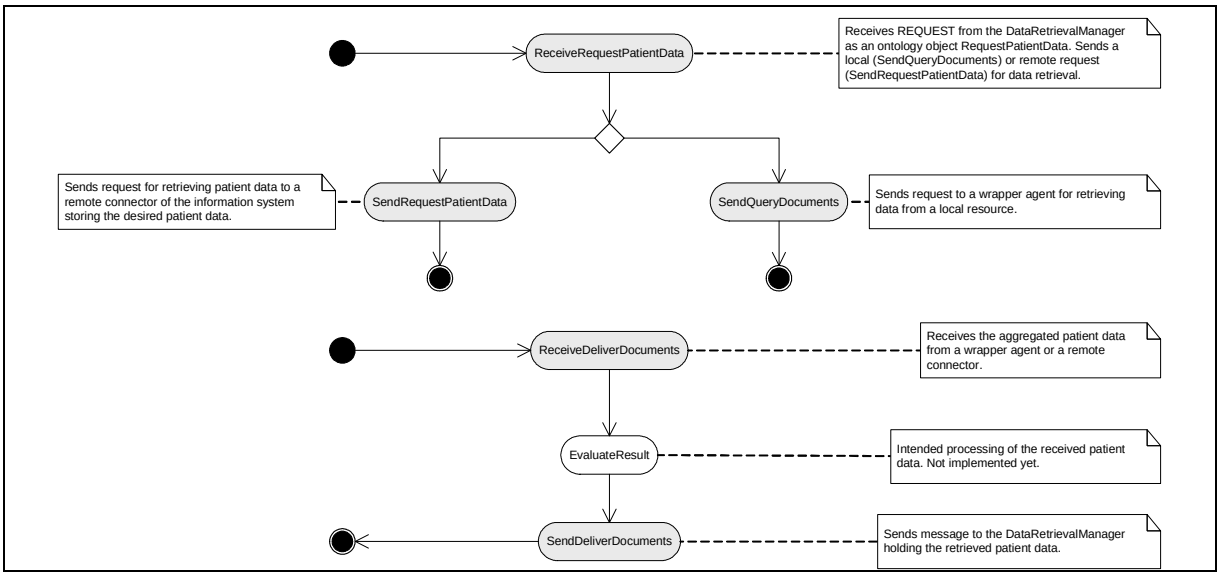

Abbildung A.5-21: Darstellung der AMDTaskAgent-Aktivität
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

A.5.12 ActiveMedicalDocument

Component ActiveMedicalDocument	
Description	Responsible for initiating all internal agents of an active, medical document: Currently, these internal agents are the DataWrapper, DataRetrievalManager, ViewManager, AppointmentMonitor, AppointmentManager, ProcessMasterAgent, and Subscriber.
Relations	**inherit** Agent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = ActionTask; InteractionTask (for a differentiation see Abbildung A.5-22); Ontology = ASAinlog; Locality = stationary;
Attributes	**private final** ReceiveOpenEHR receiveOpenEHR; **private static** InetAddress thisHost; **private static** String hostName; **protected** Agent DataWrapper; // DataRetrievalManager **protected** Agent DRManager;

Attributes	// ViewManager
	protected Agent Vmanager;
	// AppointmentMonitor
	protected Agent ApMonitor;
	// AppointmentManager
	protected Agent ApManager;
	protected Agent Subscriber;
	// References the agent controlling treatment processes
	protected Agent ProcessMasterAgent;

Operations	**private boolean** startInternalAgents **private void** sendErrorMsg **private boolean** startProcessMasterAgent **private void** killPMAgent **protected void** takeDown

Tabelle A.5-12: ***Darstellung der ActiveMedicalDocument-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

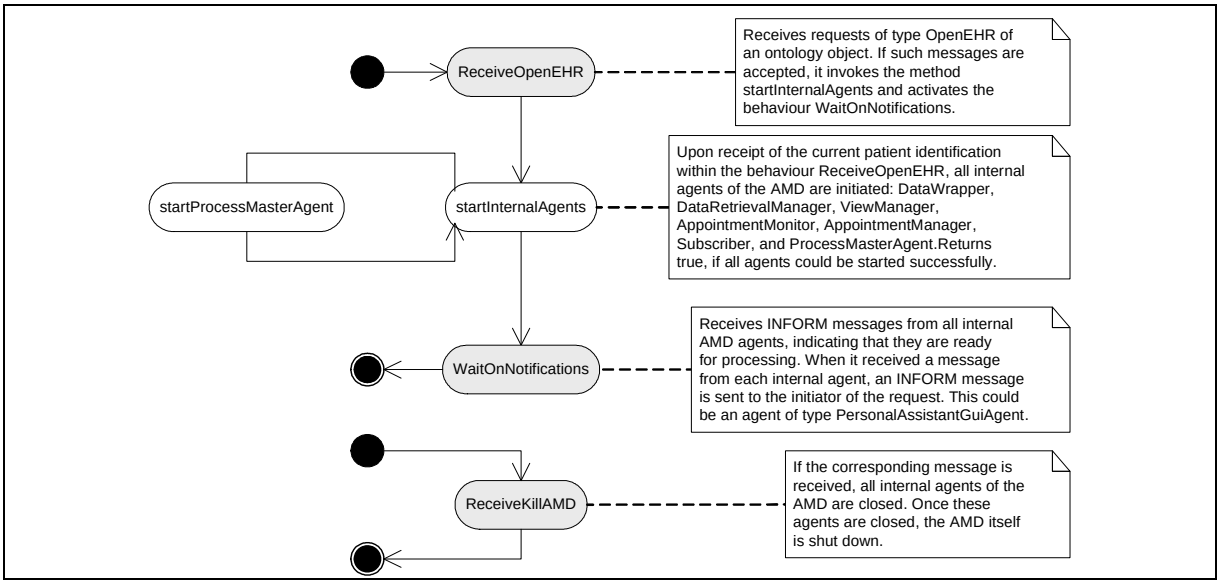

Abbildung A.5-22: ***Darstellung der ActiveMedicalDocument-Aktivität***
Quelle: Eigene Darstellung

A.5.13ResourceSchedulingTaskAgent

Component ResourceSchedulingTaskAgent	
Description	Deals with representing resources for scheduling their appointments. Receives messages and processes these.
Relations	**inherit** SchedulingTaskAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask Ontology = ASAinlog; Locality = stationary;
Attributes	**private boolean** reScheduleInit; **private boolean** durRearrange;
Operations	**protected void** setup **public** Vector rearrangeAppointments **public void** lookUpStartAffectedPatientAgents **public** AbsoluteTimepoint lastTime **public void** addReservedSchedule **public boolean** isRescheduleInit **public void** setRescheduleInit **public boolean** isDurRearrange **public void** setDurRearrange

Tabelle A.5-13: ***Darstellung der ResourceSchedulingTaskAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

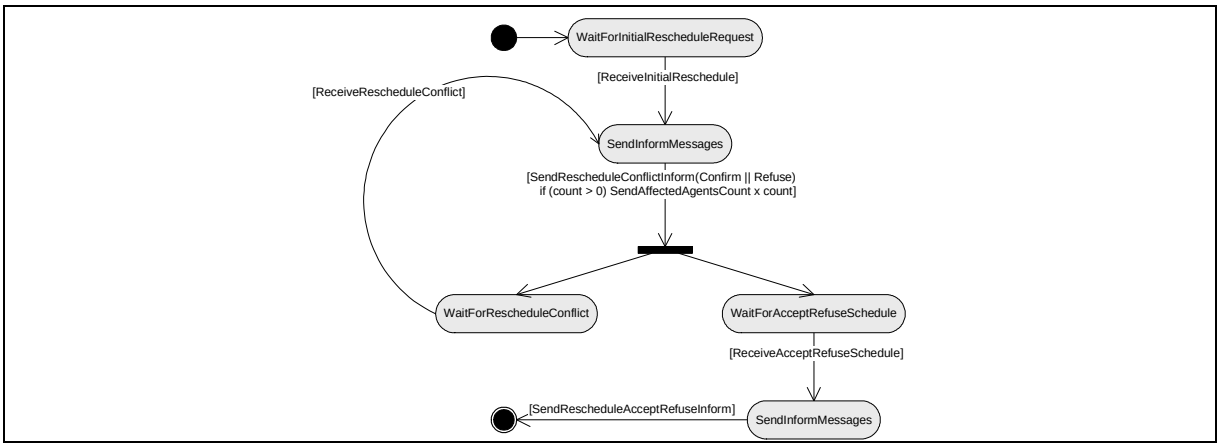

Abbildung A.5-23: Darstellung der ResourceSchedulingTaskAgent-Aktivität
Quelle: *Riedl* (2007, 56)

A.5.14 PatientSchedulingTaskAgent

Component PatientSchedulingTaskAgent	
Description	PatientSchedulingTaskAgent deals with representing patients for scheduling their appointments. Receives messages and processes these.
Relations	**inherit** SchedulingTaskAgent
AgentProfile	Lifecycle = AgentLifeCycle; TaskTypes = InteractionTask Ontology = ASAinlog; Locality = stationary;
Attributes	**private int** prio; **private** Calendar startPatientTime; **private** float costs; **private boolean** newAppointmentRescheduleToken; **private** Vector<Appointment> refusedApps; **private** Vector<Appointment> collateralRescheduleEarlier; **private** Vector<Appointment> collateralRescheduleLater;

Attributes	**private boolean** earlierToken; **private boolean** durationToken; **private boolean** scheduleUpdate;
Operations	**protected void** setup **public float** calcPrice **public float** calcCost **public float** calcNormTime **public float** calcTimeUnits **public** AbsoluteTimepoint lastTime **public** RescheduleAppointment getNextTimeslot **public** Vector<TimeInterval> orderAppointments **public** RescheduleAppointment getBestTermin **public boolean** checkConsistency **public void** checkChangedAppointment **public void** prioChanged **public void** deleteApp **public boolean** startRescheduleNewApp **public void** updateSuccPre **public void** startRescheduleLater **public void** checkSucceedingConsistency **public void** addReservedSchedule **public void** collateralReserved **public int** getPrio **public void** setPrio **public** Calendar getStartPatientTime **public void** setStartPatientTime **public float** getCosts **public void** setCosts **public void** addCosts **public boolean** isNewAppointmentRescheduleToken

Operations	**public void** setNewAppointmentRescheduleToken
	public Vector<Appointment> getRefusedApps
	public void setRefusedApps
	public void addRefusedApps
	public boolean isScheduleUpdate
	public void setScheduleUpdate
	public boolean isEarlierToken
	public void setEarlierToken
	public boolean isDurationToken
	public void setDurationToken
	public Vector<Appointment> getCollateralRescheduleEarlier
	public void addCollateralRescheduleEarlier
	public void setCollateralRescheduleEarlier
	public Appointment removeFirstCollateralEarlier
	public Vector<Appointment> getCollateralRescheduleLater
	public void setCollateralRescheduleLater
	public void addCollateralRescheduleLater
	public Appointment removeFirstCollateralLater

Tabelle A.5-14: ***Darstellung der PatientSchedulingTaskAgent-Funktionalität***
Quelle: Eigene Darstellung, abgeleitet aus der Implementierung

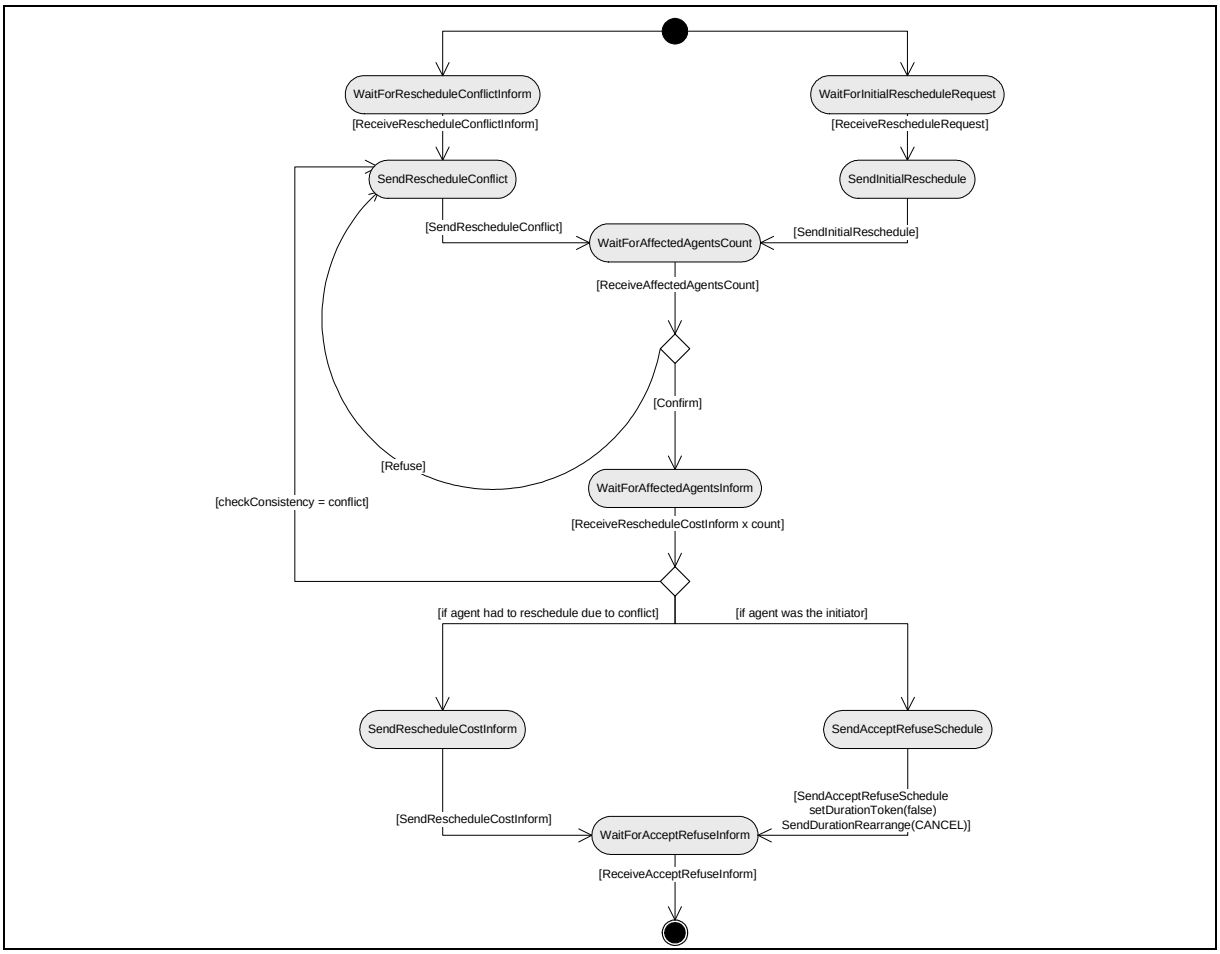

Abbildung A.5-24: Darstellung der PatientSchedulingTaskAgent-Aktivität
Quelle: *Riedl* (2007, 55)

A.6 Interaktionsdiagramme für den Scheduling-Algorithmus

A.6.1 Interaktionen für die Aktualisierung von Patiententerminen

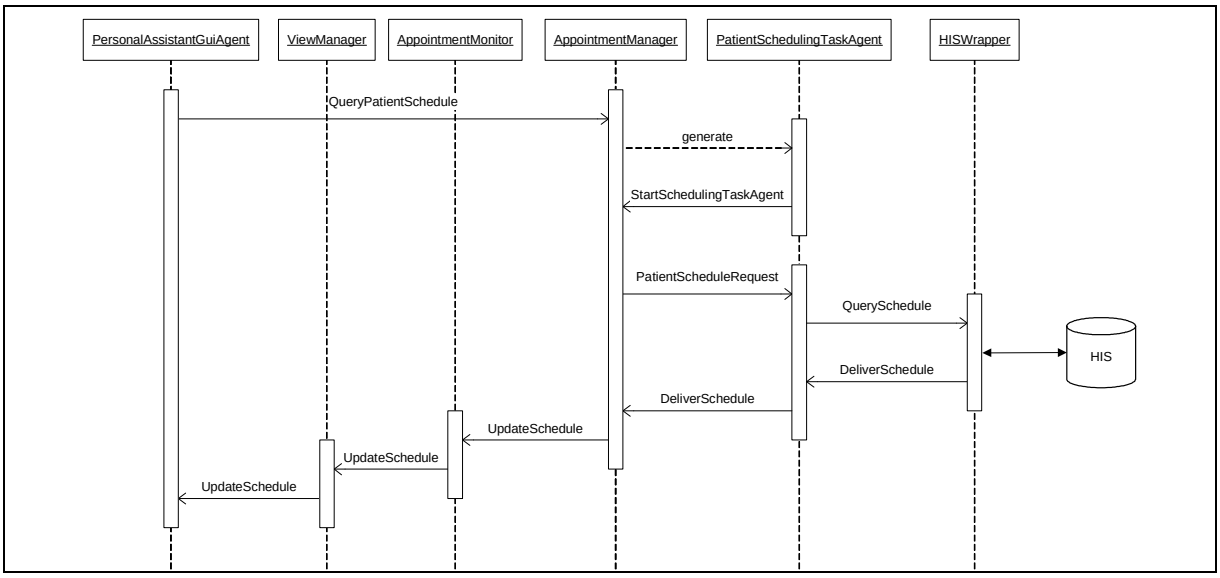

Abbildung A.6-1: ***Aktualisierung und Visualisierung von Patiententerminen***
Quelle: In Anlehnung an *Riedl* (2007, 68)

A.6.2 Interaktionen für die Aktualisierung von Ressourcen-Terminen

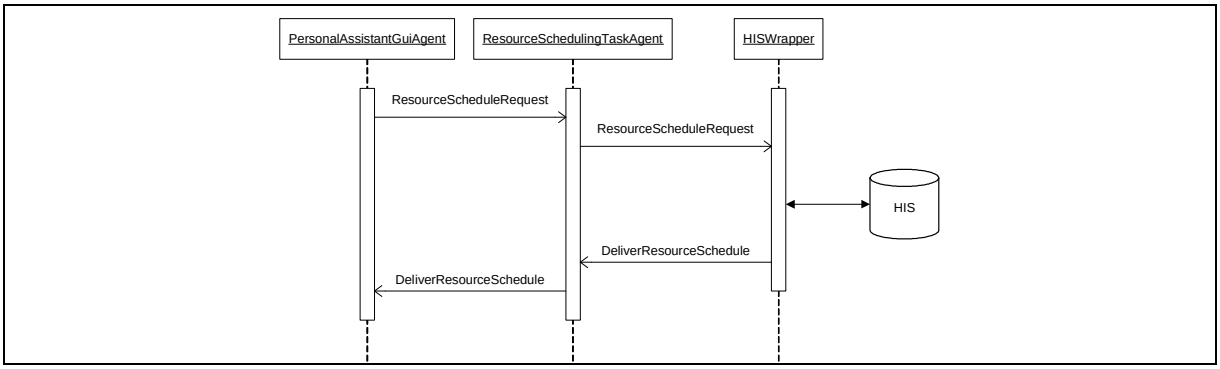

Abbildung A.6-2: ***Aktualisierung und Visualisierung von Ressourcenterminen***
Quelle: In Anlehnung an *Riedl* (2007, 68)

A.7 Aktivitätsdiagramme für den Scheduling-Algorithmus

A.7.1 Anlegen eines neuen Termins

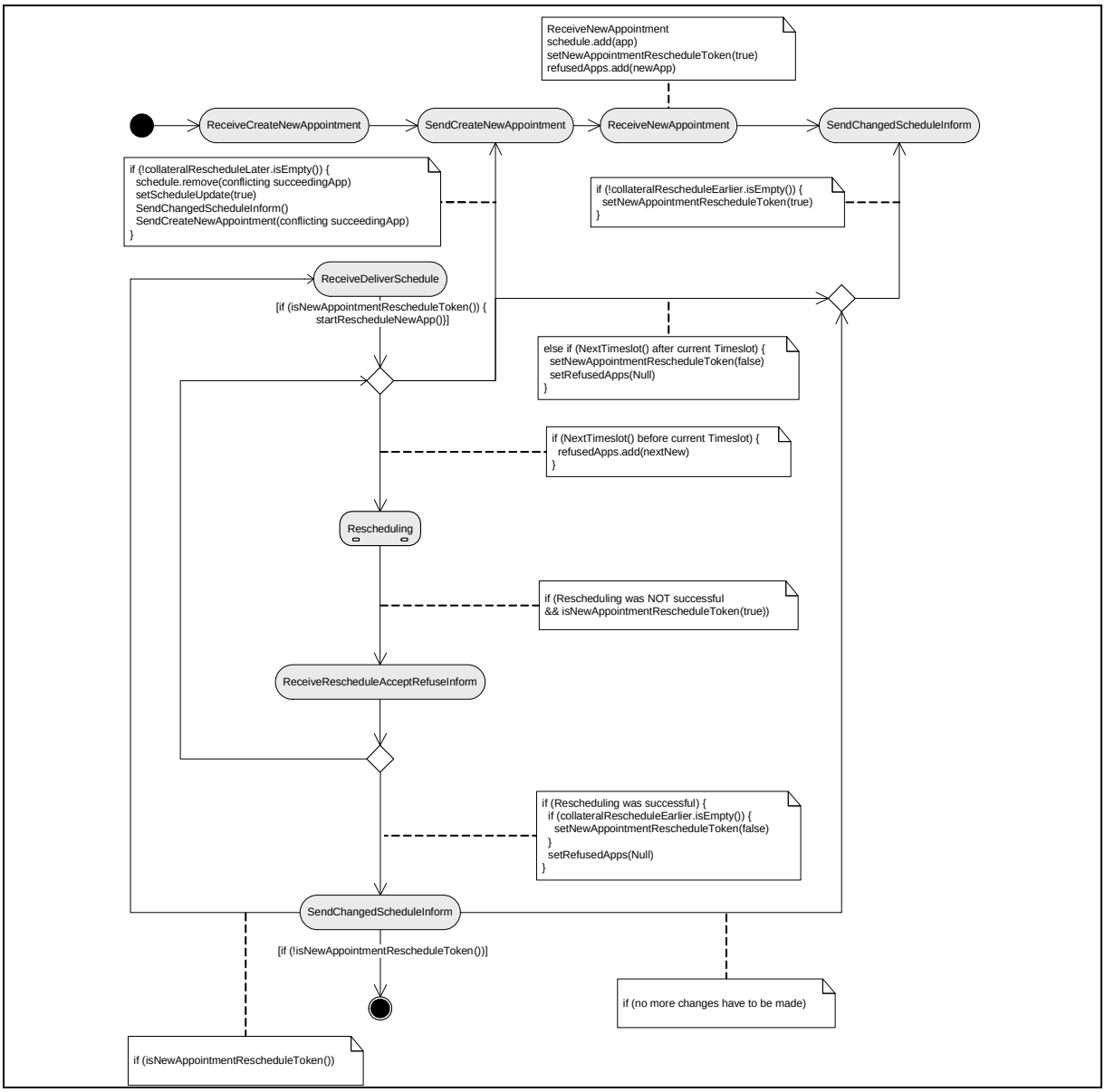

Abbildung A.7-1: Anlegen eines neuen Termins
Quelle: In Anlehnung an *Riedl* (2007, 58)

Für die in Abbildung A.7-1 dargestellten Fallunterscheidungen sind folgende Anmerkungen relevant (Riedl 2007, 58-59):

- Liste collateralRescheduleLater im Patienten-Agenten: Menge von Patiententerminen, die später anzulegen sind als der gerade betrachtete Termin; diese Termine sind beim Scheduling obligatorisch zu berücksichtigen
- Liste collateralRescheduleEarlier im Patienten-Agenten: Menge von Patiententerminen, die potenziell früher angelegt werden können als der aktuelle Termine; diese Termine können beim Scheduling fakultativ berücksichtigt werden

A.7.2 Änderung eines Termins

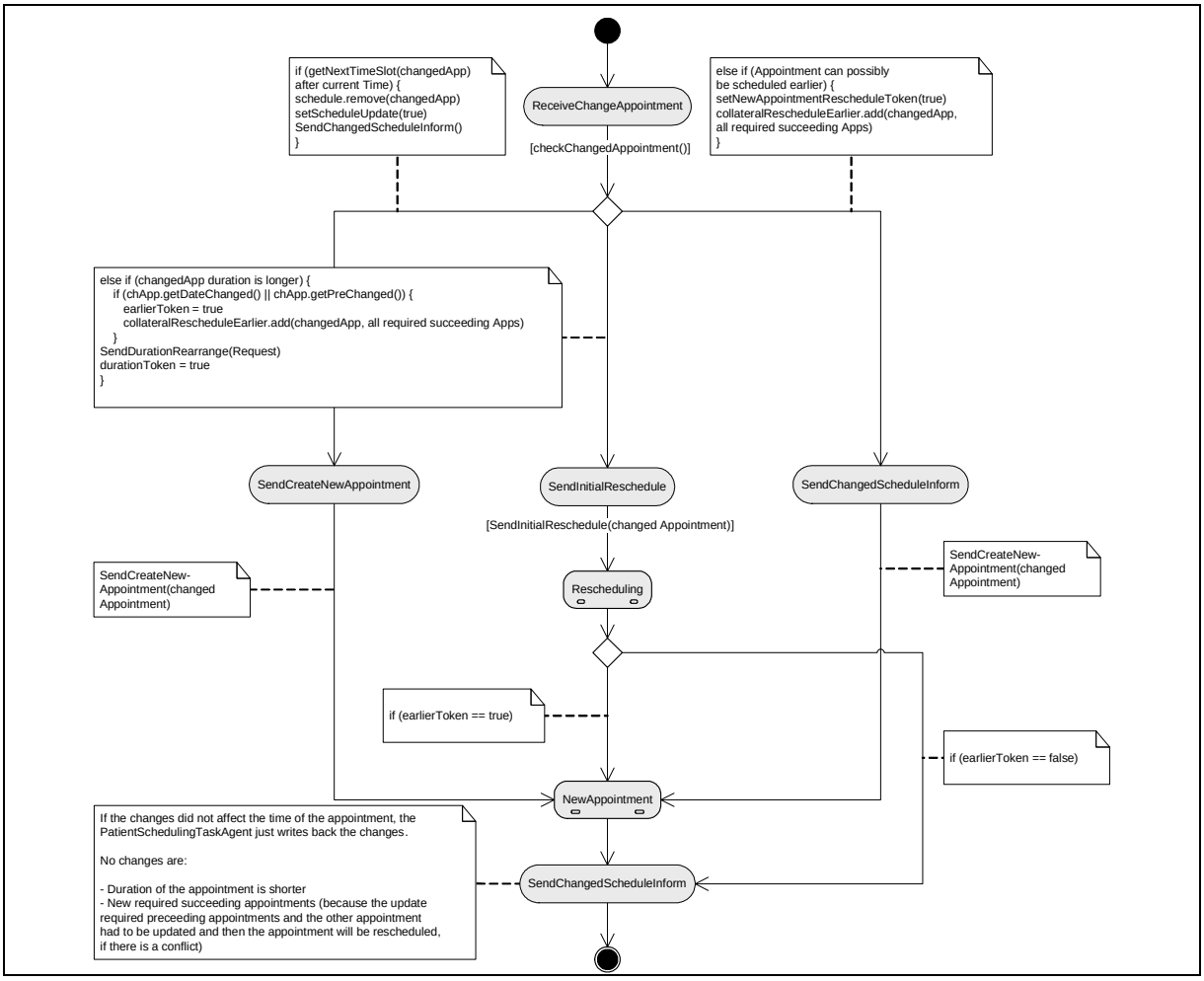

Abbildung A.7-2: ***Änderung eines Termins***
Quelle: *Riedl* (2007, 60)

A.7.3 Löschen eines Termins

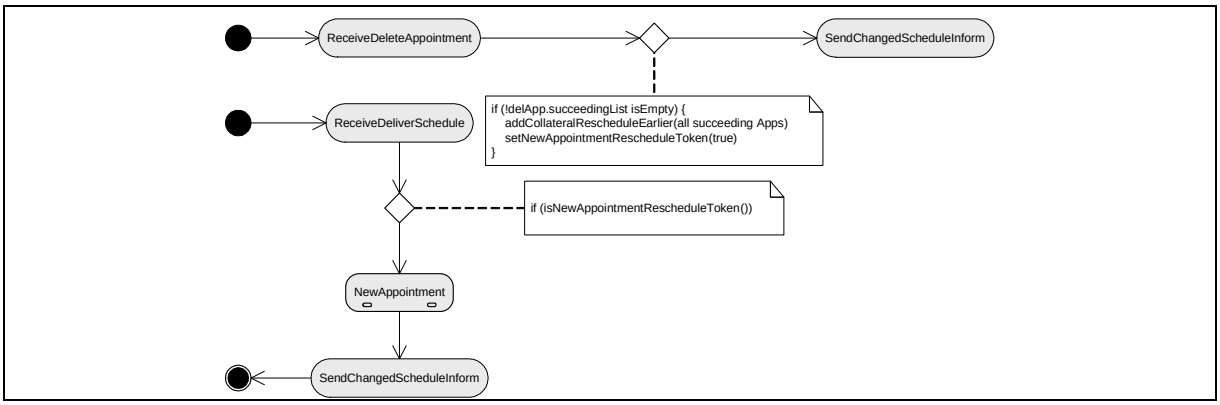

Abbildung A.7-3: ***Löschen eines Termins***
Quelle: In Anlehnung an *Riedl* (2007, 61)

A.7.4 Änderung des Gesundheitszustandes eines Patienten

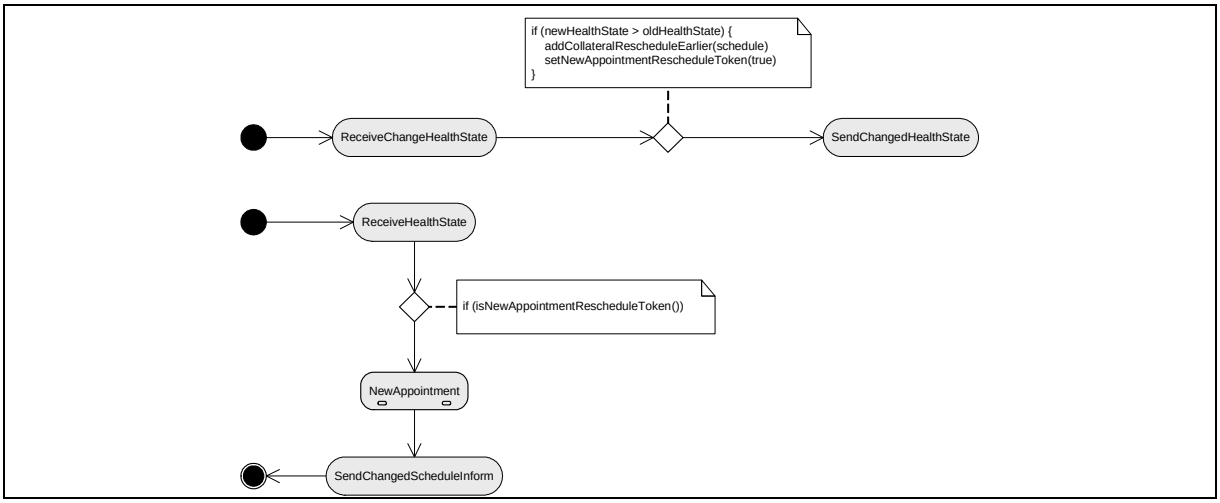

Abbildung A.7-4: ***Änderung des Gesundheitszustandes eines Patienten***
Quelle: In Anlehnung an *Riedl* (2007, 63)

A.8 Sequenzdiagramme zur Prozessunterstützung

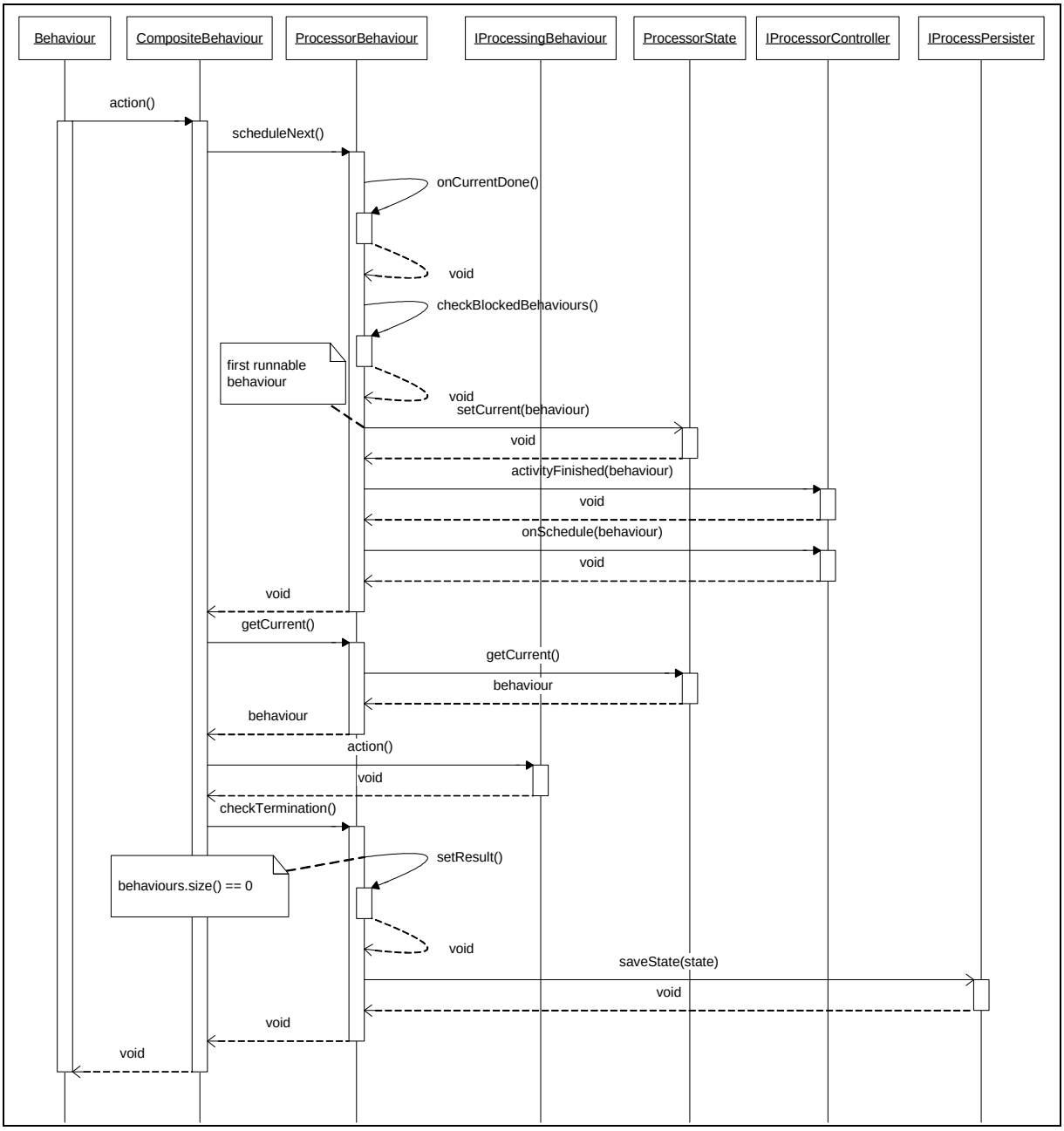

Abbildung A.8-1: ***Prozesssteuerung zur Laufzeit***
Quelle: In Anlehnung an *Bergmann/Klinger/Winkler* (2006, 63)

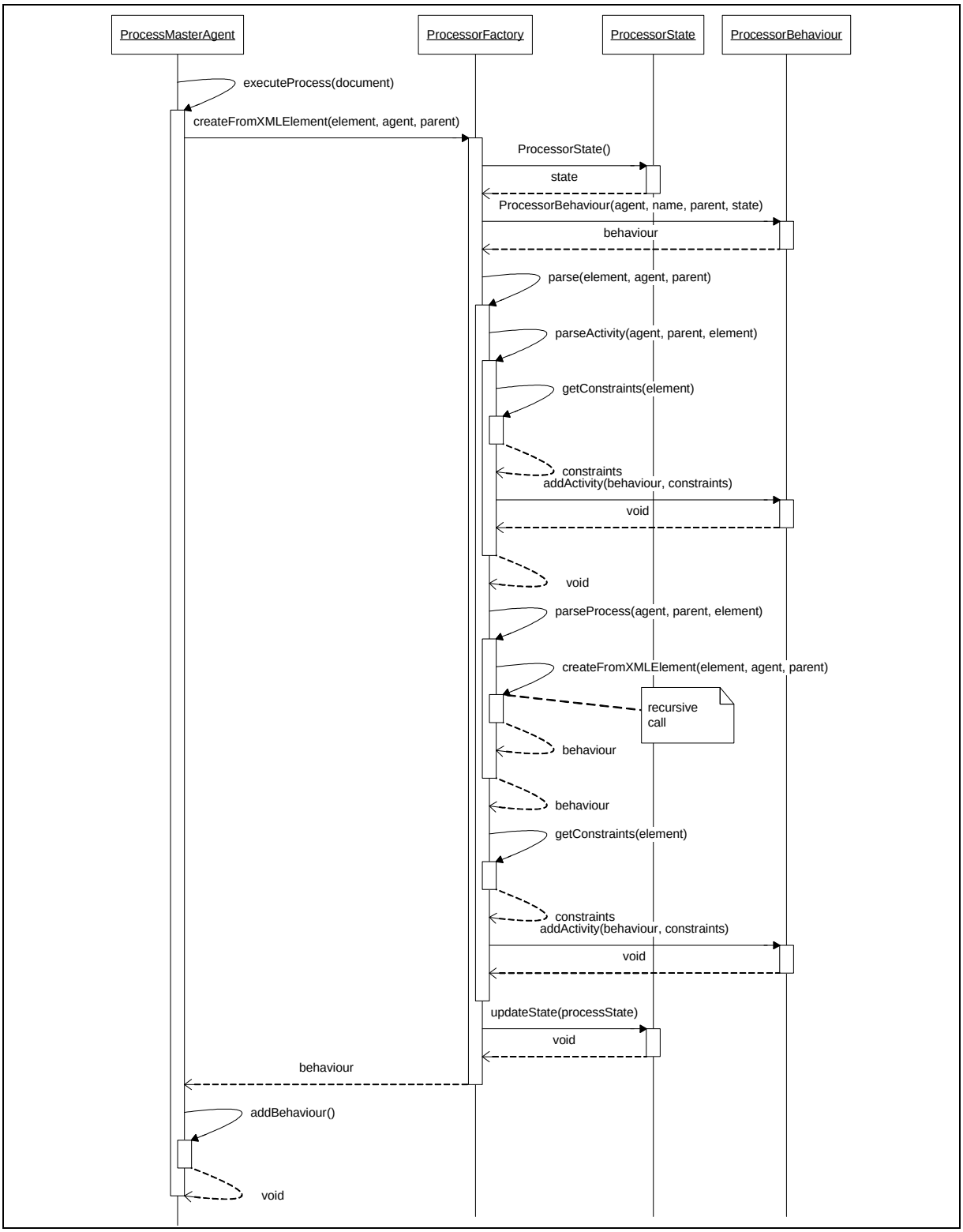

Abbildung A.8-2: Interpretationsschritte für einen abzuarbeitenden Prozess
Quelle: *Bergmann/Klinger/Winkler* (2006, 59)

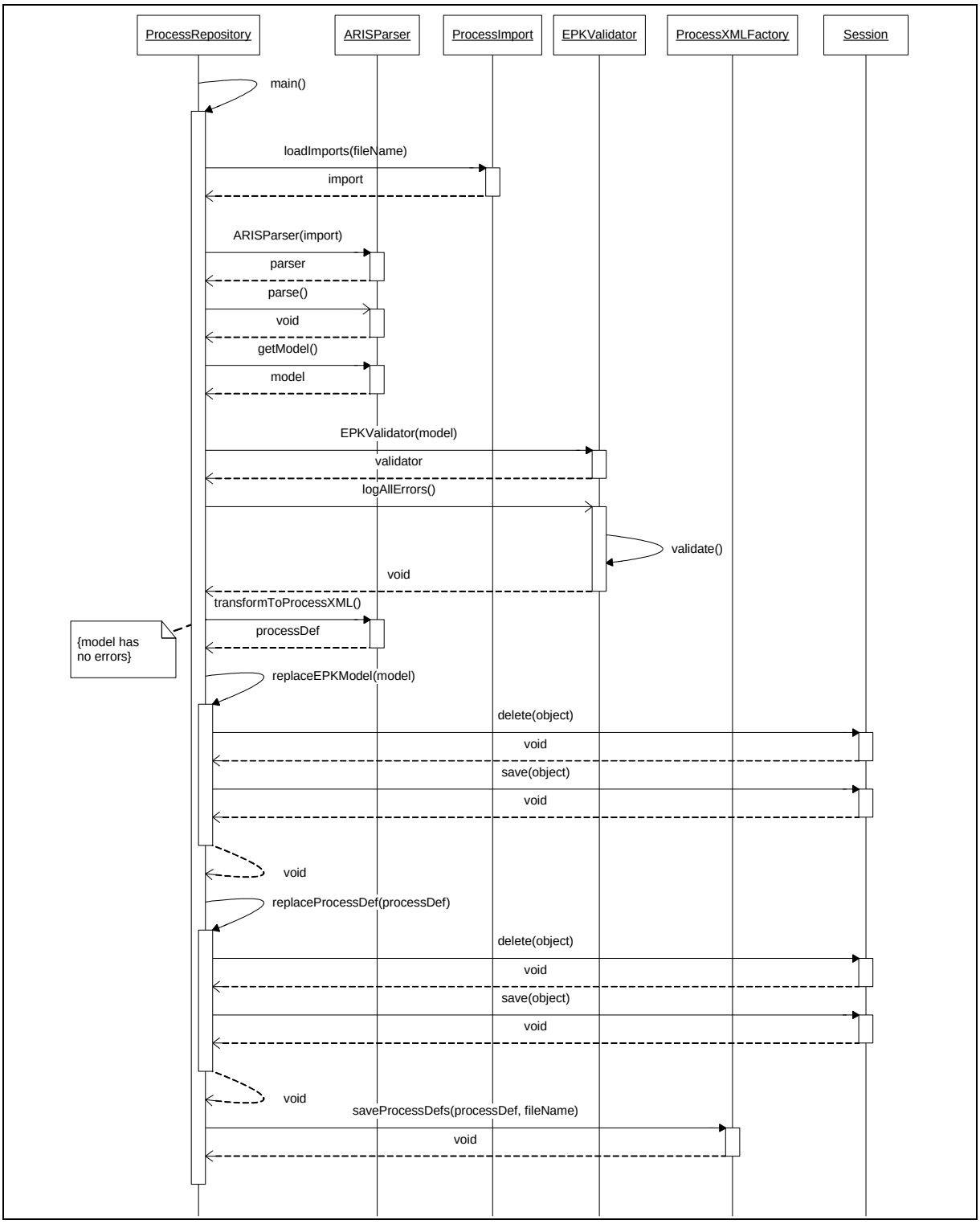

Abbildung A.8-3: Gesamtprozess für das Einlesen und Ablegen der Prozessinformation
Quelle: *Bergmann/Klinger/Winkler* (2006, 39)

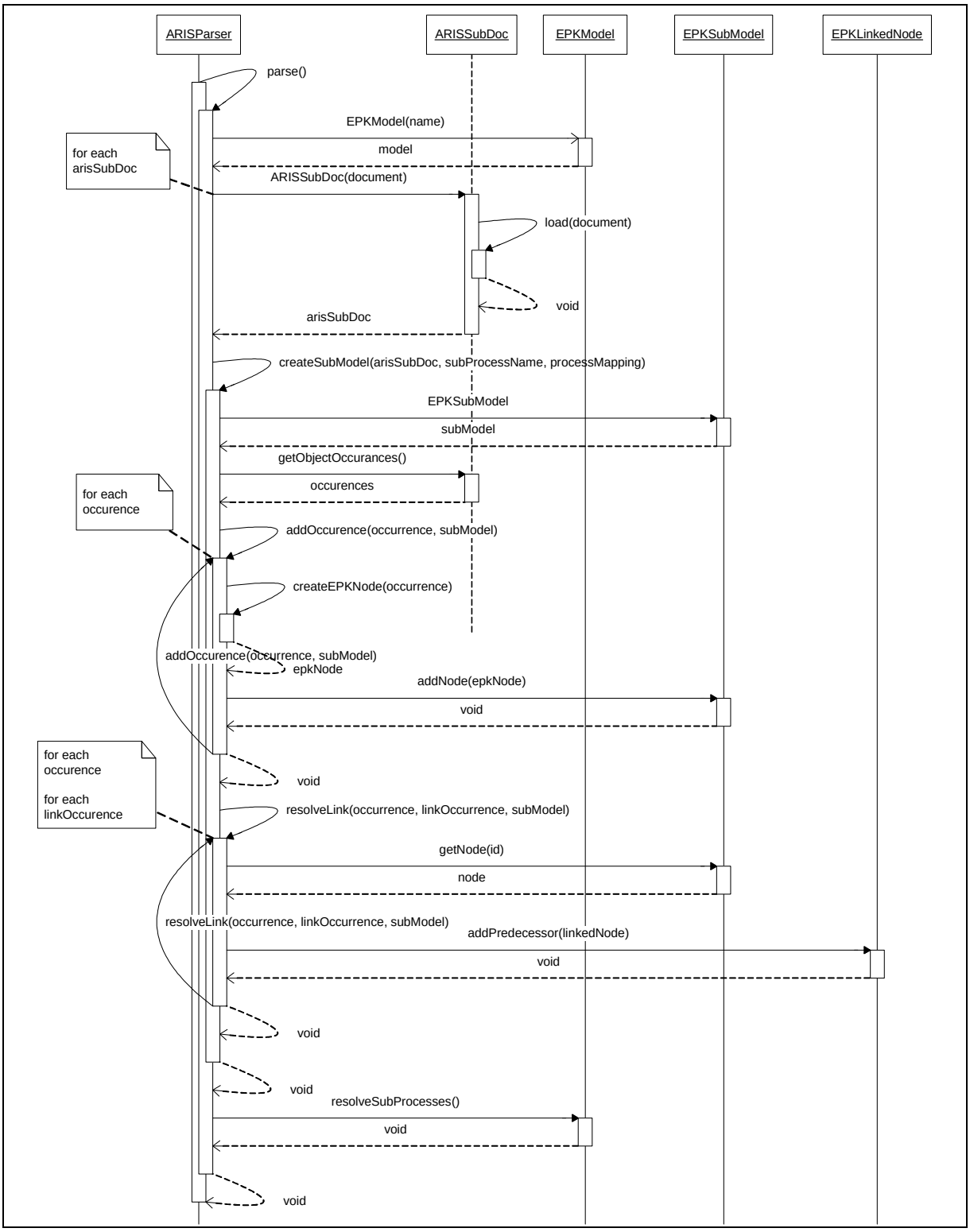

Abbildung A.8-4: ***Parse-Vorgang der AML-Beschreibung***
Quelle: *Bergmann/Klinger/Winkler* (2006, 41)

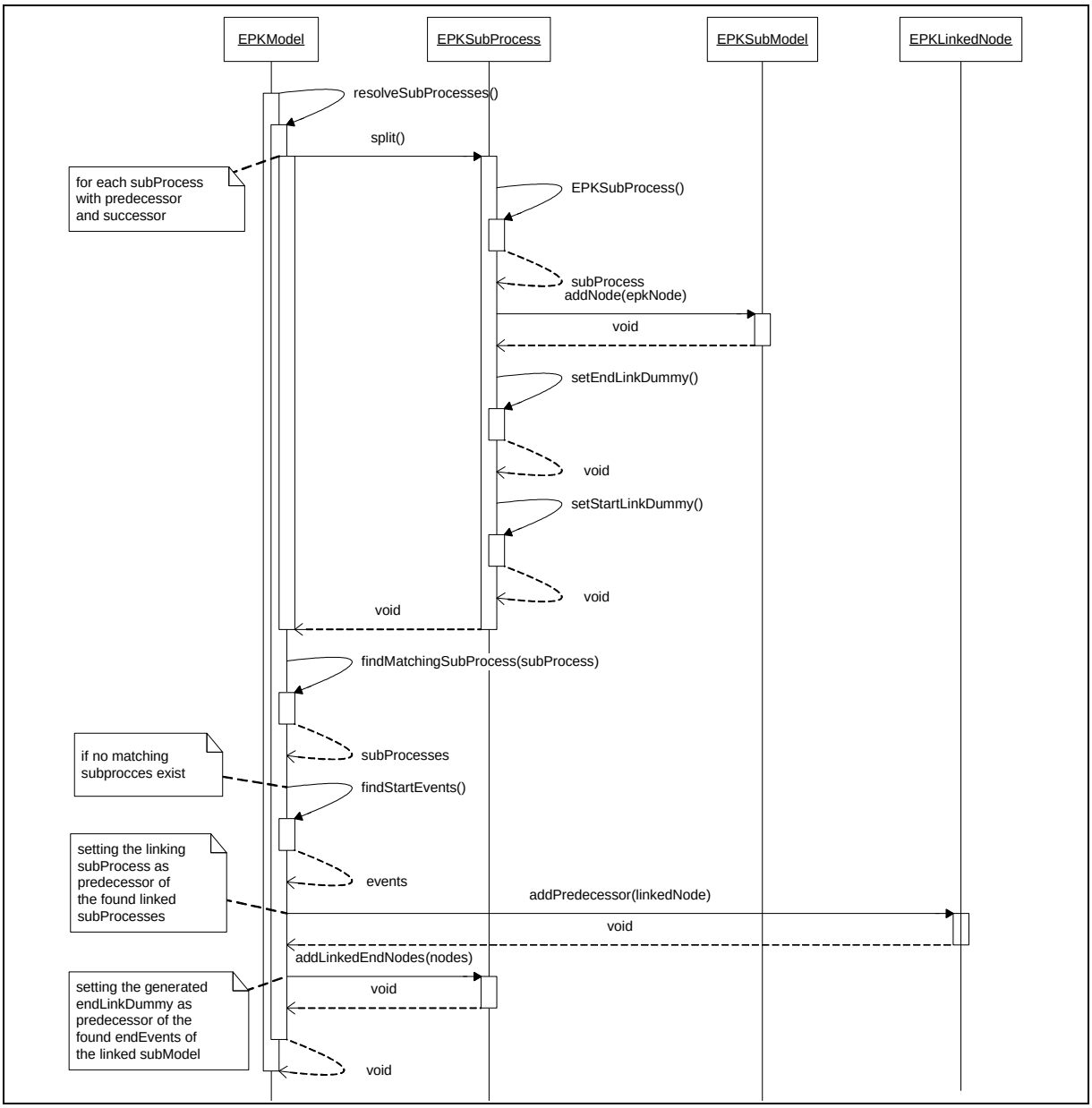

Abbildung A.8-5: Dereferenzieren von Prozesswegweisern
Quelle: In Anlehnung an *Bergmann/Klinger/Winkler* (2006, 43)

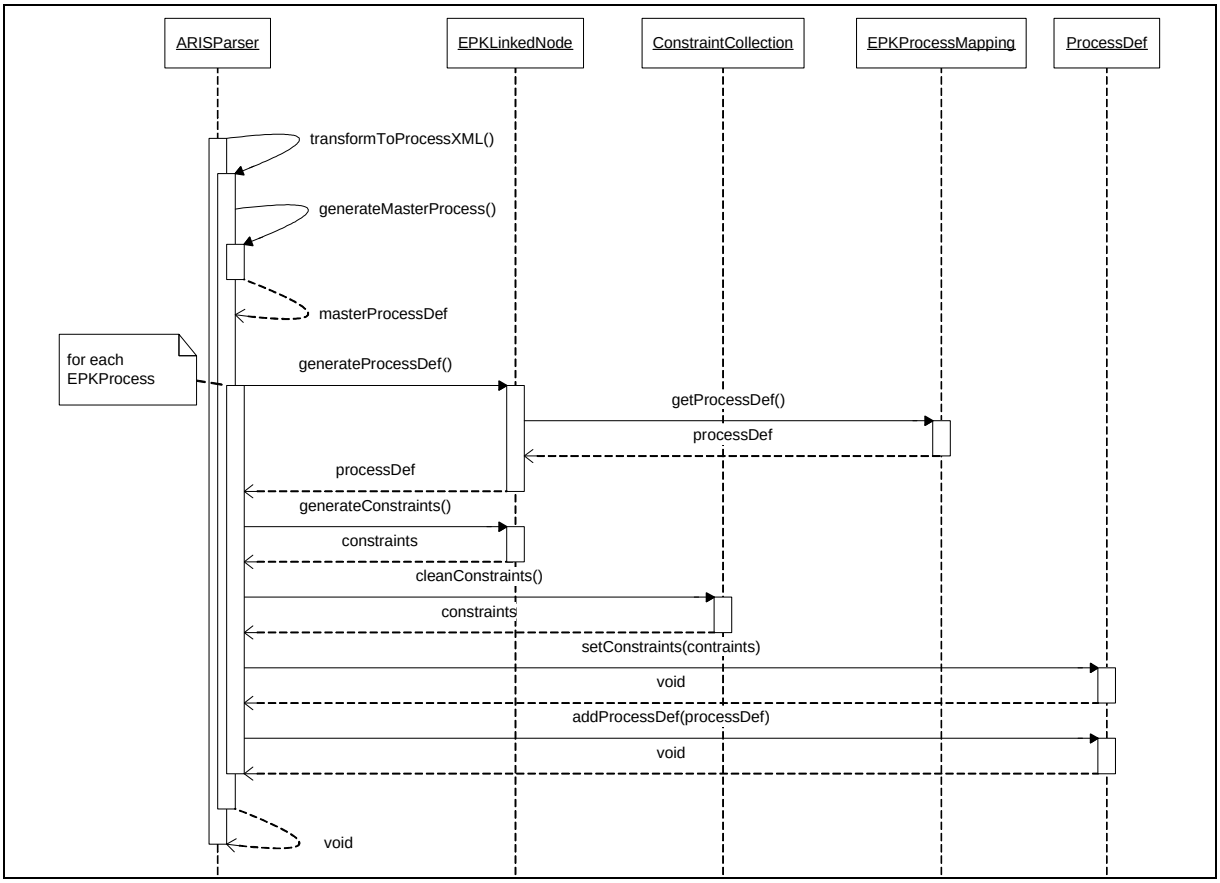

Abbildung A.8-6: Identifikation von Vorbedingungen
Quelle: *Bergmann/Klinger/Winkler* (2006, 45)

A.9 Überblick über die in der Prozesssteuerung validierten EPK-Syntaxregeln

Lfd. Nr.	Beschreibung der Syntaxregel	Fehlertyp	Klasse zur Validierung
1	Enthält die EPK mindestens ein Ereignis oder einen Prozesswegweiser ohne Vorgänger?	Fehler	EPKValidator
2	Enthält die EPK mindestens ein Ereignis oder einen Prozesswegweiser ohne Nachfolger?	Warnung	EPKValidator
3	Enthält die EPK mindestens eine Funktion?	Fehler	EPKValidator
4	Hat ein Ereignis einen Vorgänger oder einen Nachfolger?	Warnung	EPKEvent
5	Ist der Vorgänger eines Ereignisses wieder ein Ereignis?	Warnung	EPKEvent
6	Hat ein Ereignis mehr als einen Nachfolger?	Warnung	EPKEvent

7	Hat ein Operator mindestens einen Vorgänger?	Warnung	EPKOperator
8	Ist ein Operator sein eigener, direkter Vorgänger (Zirkelreferenz)?	Fehler	EPKOperator
9	Hat ein aufspaltender Operator ein Ereignis als Vorgänger?	Warnung	EPKOperator
10	Hat ein Operator einen Vorgänger?	Warnung	EPKOperator
11	Hat ein Operator einen Nachfolger?	Warnung	EPKOperator
12	Hat ein Operator mehr als einen Vorgänger und mehr als einen Nachfolger?	Warnung	EPKOperator
13	Hat eine Funktion einen Vorgänger?	Warnung	EPKProcess
14	Hat eine Funktion einen Nachfolger?	Warnung	EPKProcess
15	Hat eine Funktion mehr als einen Nachfolger?	Warnung	EPKProcess
16	Ist ein Prozesswegweiser sein eigener, direkter Vorgänger (Zirkelreferenz)?	Fehler	EPKSubProcess
17	Hat ein Prozesswegweiser einen Vorgänger oder einen Nachfolger?	Warnung	EPKSubProcess
18	Hat ein Prozesswegweiser mehr als einen Nachfolger?	Warnung	EPKSubProcess

Tabelle A.9-1: ***Darstellung der zu überprüfenden EPK-Eigenschaften***
Quelle: *Bergmann/Klinger/Winkler* (2006, 50)

EPK-Regel	**Abdeckung in der Implementierung (siehe die lfd. Nr. in Tabelle A.9-1)**
Eine EPK besteht nur aus [..] Ereignissen, Funktionen, Prozesswegweisern, Operatoren, Organisationseinheiten, Informationsobjekten.	Konzeptionell abgedeckt: es [!] werden nur Ereignisse, Funktionen, Prozesswegweiser, Operatoren, Organisationseinheiten und Informationsobjekte verarbeitet.
Verknüpfungsoperatoren müssen mit Symbol oder Verknüpfungsoperator verbunden sein.	10, 11
Eine EPK kann nur mit einem Ereignis oder einem Prozesswegweiser beginnen oder enden.	1, 2
Eine EPK muss mindestens eine Funktion enthalten.	3
Alle Symbole müssen mit Linien verbunden sein.	4, 10, 11, 13, 14, 17
Ein Ereignis kann einem anderen Ereignis nicht direkt folgen oder vorausgehen.	5
Ein Ereignis folgt oder geht einer Funktion voraus.	Standardmäßig nicht aktiv, da zu restriktiv
Ein Ereignis kann nur eine Eingangs- und eine Ausgangslinie haben.	6

Eine Funktion kann einem Prozesswegweiser oder einer anderen Funktion nicht direkt folgen oder vorausgehen.	Standardmäßig nicht aktiv, da zu restriktiv
Eine Funktion muss mindestens vor einem Ereignis stehen oder mindestens einem Ereignis folgen.	Standardmäßig nicht aktiv, da zu restriktiv
Eine Funktion kann nur eine Eingangs- und eine Ausgangslinie haben.	15
Ein Prozesswegweiser muss entweder vor einem Ereignis stehen oder wenigstens einem Ereignis folgen.	Standardmäßig nicht aktiv, da zu restriktiv
Ein Prozesswegweiser kann nicht vor einer Funktion oder einem anderen Prozesswegweiser stehen.	Standardmäßig nicht aktiv, da zu restriktiv
Ein Prozesswegweiser kann nur eine Eingangs- und eine Ausgangslinie haben.	18
Aufspaltung: Verknüpfungsoperatoren müssen einen eingehenden Pfeil und zwei oder mehrere ausgehende Pfeile haben.	7, 12
Zusammenführung: Verknüpfungsoperatoren müssen zwei oder mehr eingehende Pfeile und einen ausgehenden Pfeil haben.	7, 12
„OR“ oder „XOR“ Verknüpfungsoperatoren, die eine Gabelung in der Prozesskette darstellen, dürfen nicht einem Ereignis folgen.	9

Tabelle A.9-2: ***Zusammenhang zwischen den EPK-Regeln und ihrer Abdeckung in der prototypischen Implementierung***
Quelle: *Bergmann/Klinger/Winkler* (2006, 46)

A.10 Datenbankschemata der Prozesssteuerung

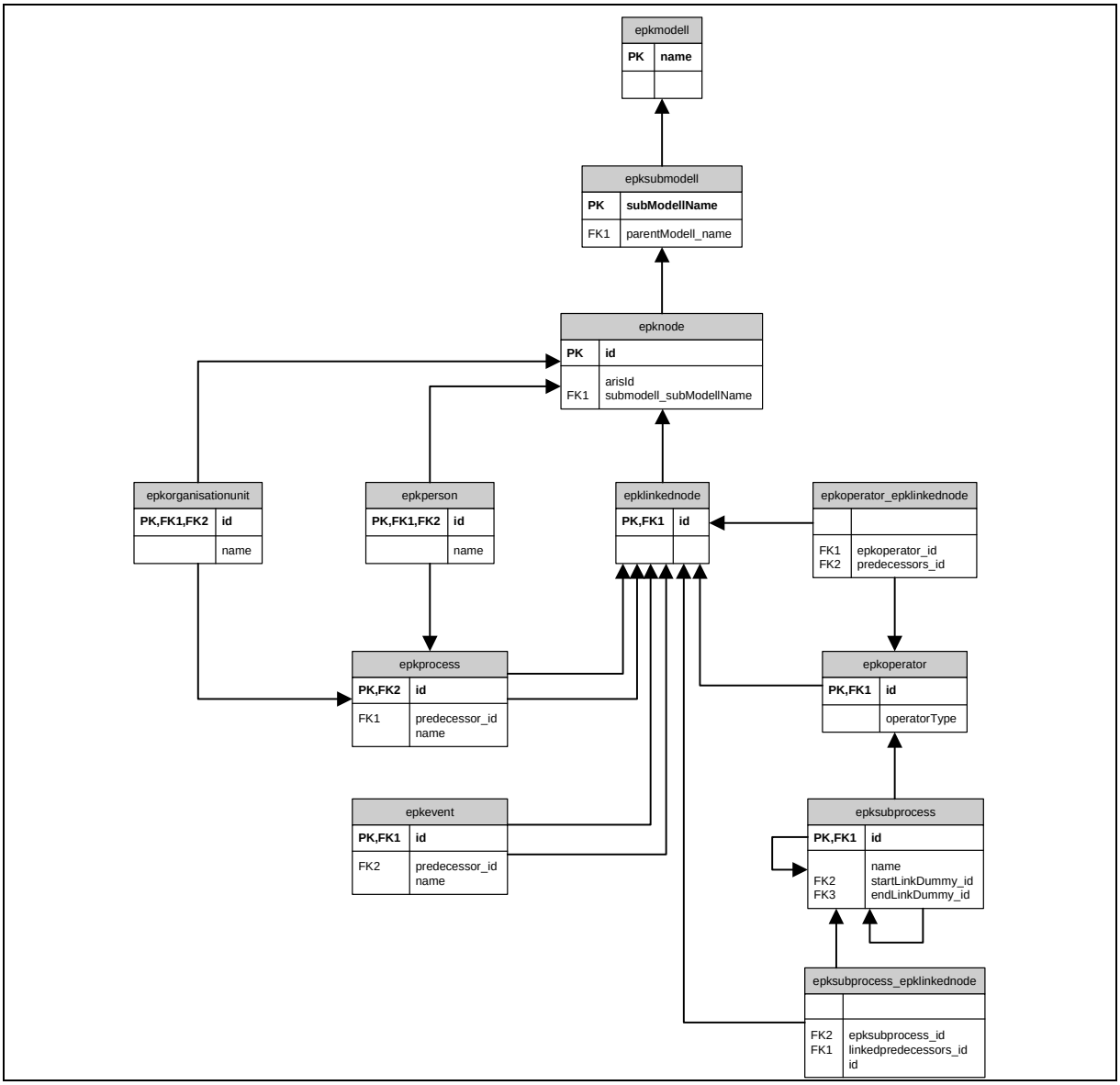

Abbildung A.10-1: Relationales Datenbankschema für das persistente EPK-Modell
Quelle: Bergmann/Klinger/Winkler (2006, 70)

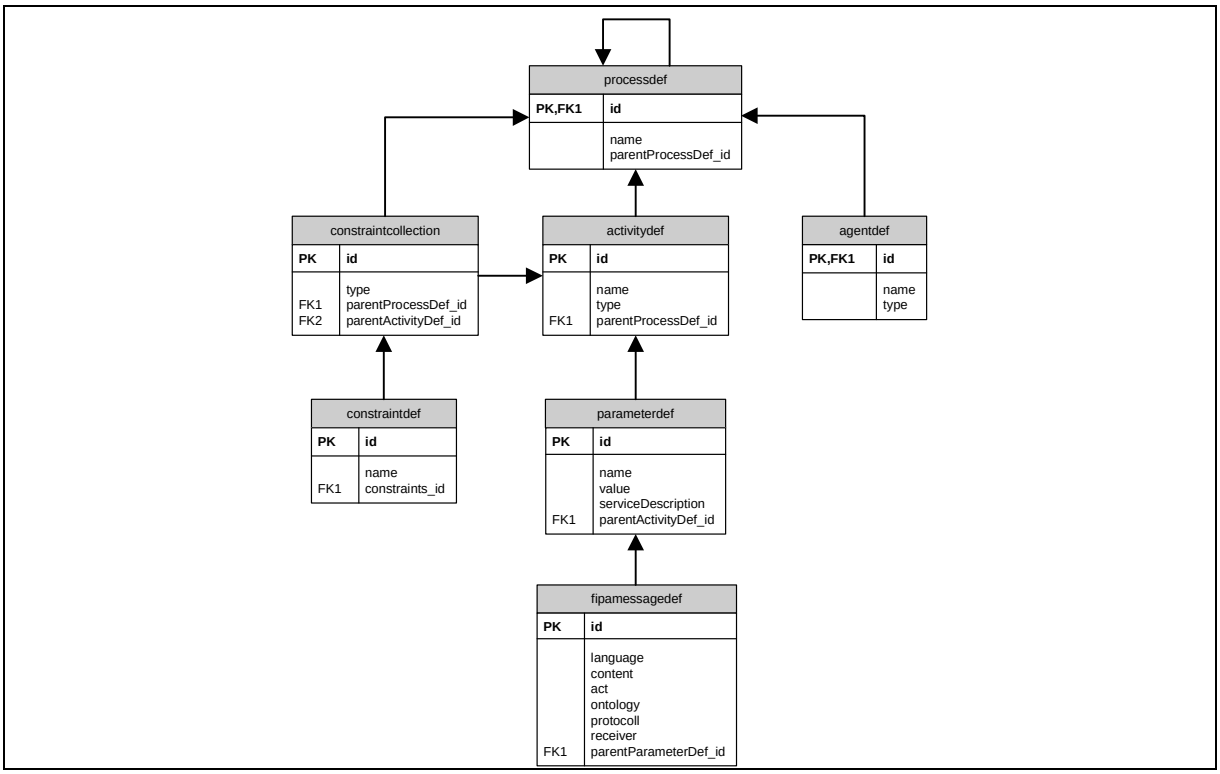

Abbildung A.10-2: Relationales Datenbankschema für das persistente PML-Modell
Quelle: *Bergmann/Klinger/Winkler* (2006, 71)

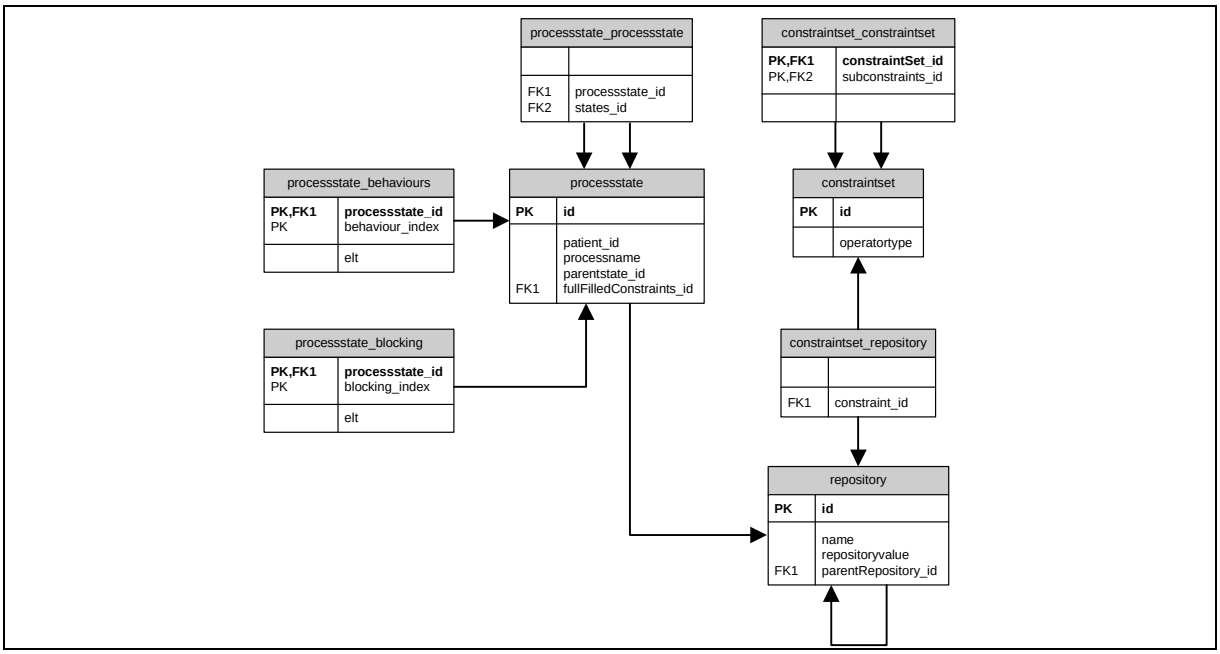

Abbildung A.10-3: Relationales Datenbankschema für das persistente ProcessState-Modell
Quelle: *Bergmann/Klinger/Winkler* (2006, 72)

A.11 Installationsanleitung

A.11.1Voraussetzungen

Für die Installation der in der vorliegenden Arbeit beschriebenen, prototypischen Implementierung werden die folgenden Rahmenbedingungen vorausgesetzt, auf deren Konfiguration im Folgenden nicht weiter eingegangen wird:

- PC mit dem Betriebssystem Windows XP Service Pack 2
- Java-Laufzeitumgebung, Version 1.5.0[107]
- Apache Ant[108]
- PDA vom Typ XDA II mit dem Betriebssystem Windows Mobile 2003

Weiterhin ist das auf dem zur vorliegenden Arbeit gehörenden Datenträger verfügbare Verzeichnis prototype[109] in ein beliebiges Verzeichnis ohne Leerzeichen auf den Installationsrechner zu kopieren. Um möglichst wenig Pfadeinstellungen vorzunehmen, wird empfohlen, das genannte Verzeichnis im Wurzelverzeichnis auf dem Laufwerk C: zu installieren.

---

[107] Aufgrund der langen Entwicklungszeit der in der vorliegenden Arbeit beschriebenen Implementierung wurden neuere Versionen der Java-Laufzeitumgebung nicht berücksichtigt. Deshalb ist bei der Installation darauf zu achten, die genannte Version einzusetzen, um die korrekte Ausführung der gesamten Anwendung zu gewährleisten.

[108] Apache Ant ist auf der folgenden Seite verfügbar: http://ant.apache.org/index.html, zugegriffen am 08.09.2007.

[109] In diesem Verzeichnis befinden sich alle zu der gesamten Anwendung gehörenden Quell- und Objektdateien. Insbesondere sind darin auch die zum Medizinischen Informationssystem gehörenden Dateien enthalten, weshalb in Abschnitt A.11 keine separaten Ausführungen zur Installation der Desktop-Anwendung beschrieben sind.

Die für die Anwendungen Prozesssteuerung und Patientenportal eingesetzte Datenbank MySQL wird so installiert und konfiguriert (für die Beschreibung der Installation siehe auch Bergmann/Klinger/Winkler 2006, 94-95), wie in Abbildung A.11-1 mit Abbildung A.11-6 beschrieben ist. Dabei können meist die Standardeinstellungen übernommen werden. Wichtig für die korrekte Ausführung der gesamten Anwendung ist die Verwendung des root-Passwortes root (siehe Abbildung A.11-6 und Abschnitt A.11.2).

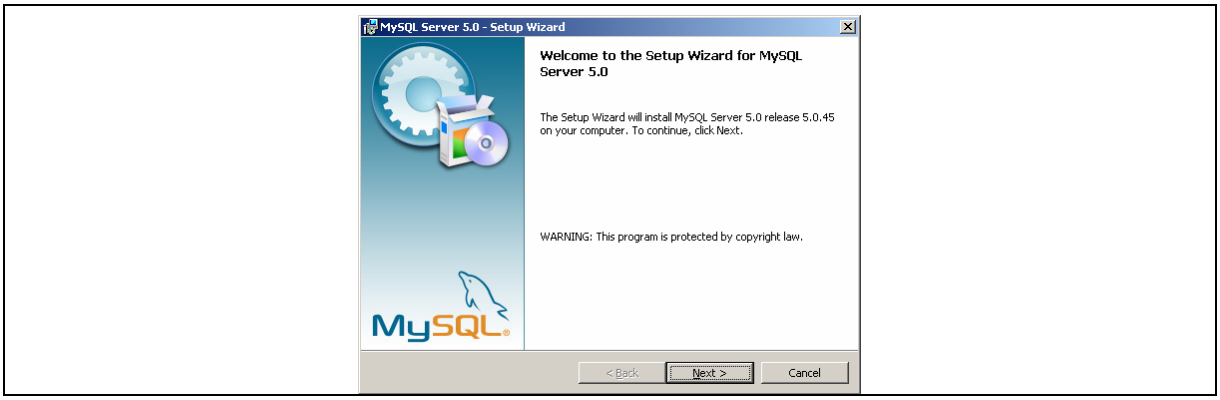

Abbildung A.11-1: ***Start der Installation der Datenbank***
Quelle: Screenshot zur Installation der Datenbank MySQL

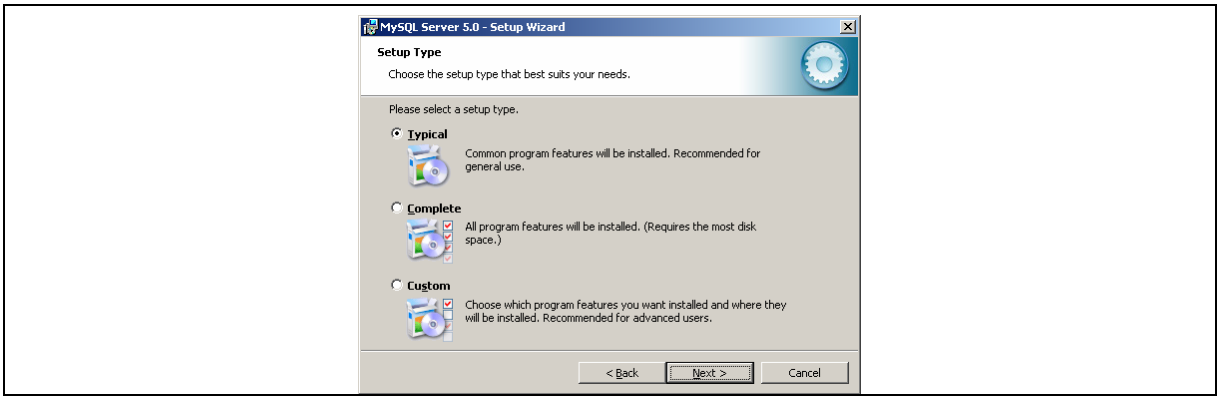

Abbildung A.11-2: ***Auswahl des Installationsmodus***
Quelle: Screenshot zur Installation der Datenbank MySQL

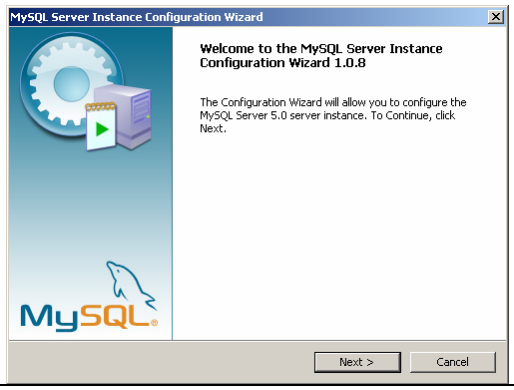

Abbildung A.11-3: ***Start der Konfiguration der Datenbank***
Quelle: Screenshot zur Konfiguration der Datenbank MySQL

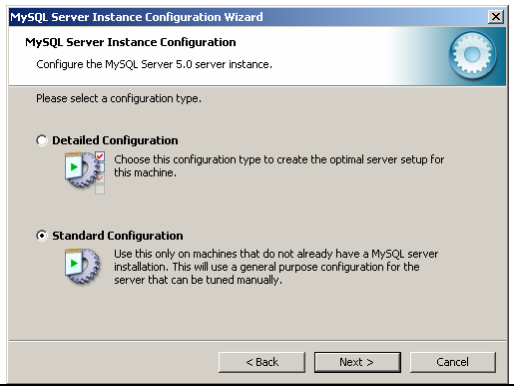

Abbildung A.11-4: ***Auswahl der Standardkonfiguration für die Datenbank***
Quelle: Screenshot zur Konfiguration der Datenbank MySQL

Abbildung A.11-5: ***Spezifikationen von Optionen***
Quelle: Screenshot zur Konfiguration der Datenbank MySQL

Abbildung A.11-6: ***Festlegen des root-Passworts:*** root
Quelle: Screenshot zur Konfiguration der Datenbank MySQL

Zur Administration der Datenbank wird für die folgende Beschreibung das Werkzeug MySQL Tools eingesetzt. Dessen Installation wird in Abbildung A.11-7 mit Abbildung A.11-9 beschrieben.

Abbildung A.11-7: ***Start der Installation der Datenbankwerkzeugsammlung***
Quelle: Screenshot zur Installation der Applikation MySQL Tools

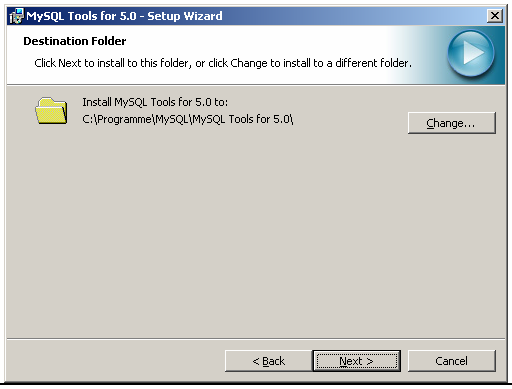

Abbildung A.11-8: ***Auswahl des Installationsverzeichnisses für die Werkzeugsammlung***
Quelle: Screenshot zur Installation der Applikation MySQL Tools

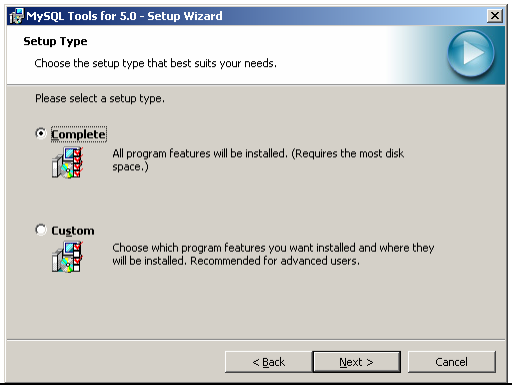

Abbildung A.11-9: ***Auswahl des Installationsmodus***
Quelle: Screenshot zur Installation der Applikation MySQL Tools

A.11.2Überblick über Passwörter und Benutzer

In Tabelle A.11-1 ist ein Überblick über die in der prototypischen Implementierung angelegten Benutzer sowie ihre jeweiligen Passwörter bzw. PINn dargestellt. Für Benutzer in der Rolle des Patienten spezifiziert der zweite Aufzählungspunkt in der ersten Spalte den Login-Namen für das Portal. Für die Datenbankbenutzer wird das Passwort jeweils in Spalte „Passwort Medizinisches Informationssystem“ angegeben.

Name	Rolle	Passwort Medizinisches Informationssystem	PIN eGK bzw. HBA	Passwort Portal
asainlog	Datenbankbenutzer	asainlog	Nicht zutreffend	Nicht zutreffend
root	Datenbankbenutzer	root	Nicht zutreffend	Nicht zutreffend
• Gustav Kaiser • kaiser@liferay.com	Patient	Nicht erforderlich	1111	1111
• Maximilian Schneider • schneider@liferay.com	Patient	Nicht erforderlich	1111	1111
• Alexander Neumann • neumann@liferay.com	Patient	Nicht erforderlich	1111	1111
• Franz Mohrmann • mohrmann@liferay.com	Patient	Nicht erforderlich	1111	1111
meier	Anwender	Leeres Passwort	1111	Nicht erforderlich
klinger	Anwender	Leeres Passwort	1111	Nicht erforderlich
obermann	Anwender	Leeres Passwort	1111	Nicht erforderlich
neumeier	Anwender	Leeres Passwort	1111	Nicht erforderlich

Tabelle A.11-1: ***Überblick über die in der prototypischen Implementierung angelegten Benutzer und die zugehörigen Passwörter***
Quelle: Eigene Darstellung

A.11.3Vorbereitung der mobilen Anwendung

Im Folgenden wird beschrieben, welche Schritte für die Installation der mobilen Anwendung erforderlich sind (siehe dazu auch Hillebrand 2006, 105-111):

In der Datei de.tum.winfo.asainlog.mobile.Environment.java im Verzeichnis prototype\asainlog\src ist die Variable targetHostName mit dem Namen des stationären Rechners zu belegen. Für eine Übernahme dieser Einstellung für die mobile Anwendung ist der folgende Kommandozeilenbefehl auszuführen:

- ant mobile-do-jar im Verzeichnis prototype\asainlog

Bei der Installation von J2ME Wireless Toolkit auf dem stationären Rechner können alle Standardeinstellungen übernommen werden. Um eine Konfiguration der Datei build.xml im Verzeichnis prototype\asainlog zu umgehen, empfiehlt sich die Installation in dem Pfad C:\Programme\WTK22. Wenn für die Installation ein anderes Verzeichnis gewählt wurde, ist die Variable j2mewtk.home in der Datei build.xml im genannten Verzeichnis mit demjenigen Wert zu belegen, der dem Pfad für J2ME Wireless Toolkit entspricht.

Für die Installation von IBM WebSphere Everyplace Micro Environment auf dem stationären Rechner bzw. den PDA ist eine Verbindung zwischen diesen beiden Geräten über ActiveSync® aufzubauen. Nach dem Verbindungsaufbau können bei der Installation alle Standardeinstellungen übernommen werden.

Auf dem PDA ist ein geeignetes Verzeichnis anzulegen, in das die Dateien für die mobile Applikation kopiert werden. Um die Anpassung von Konfigurationsdateien zu vermeiden, wird empfohlen, das Verzeichnis asainlog im Wurzelverzeichnis des PDAs anzulegen. In dieses Verzeichnis werden über eine mit ActiveSync etablierte Verbindung zwischen dem Desktop-Rechner und dem PDA die Dateien ASAinlogMobile.jar, ASAinlogMobile.lnk und settings.conf aus dem Verzeichnis prototype\asainlog\build\build-mobile\bin kopiert. Die Datei ASAinlog-Mobile.lnk ist ggf. anzupassen, falls auf dem PDA ein anderes Installationsverzeichnis für die mobile Anwendung gewählt wurde. In der Datei settings.conf ist ggf. eine Anpassung der IP-Adresse des stationären Rechners erforderlich, unter der letzterer in der Verbindung für den Datenaustausch zwischen BackEnd und FrontEnd erreichbar ist.

Auf dem stationären Rechner ist für diejenige Netzwerkverdingung, über die die Kommunikation zwischen FrontEnd und BackEnd von JADE-LEAP erfolgt, die IP-Adresse 192.168.100.1 zu konfigurieren. Für den PDA ist die Konfiguration für die IP-Adresse 192.168.100.2 vorzunehmen.

A.11.4Vorbereitung der Portalanwendung

Die Vorbereitung der Portalanwendung wird in diesem Abschnitt beschrieben. Zunächst ist das auf dem Datenträger im Verzeichnis apps befindliche Verzeichnis apache-tomcat-5.5.16-jade auf dem Zielrechner in das Verzeichnis C:\Programme zu kopieren. Wenn ein anderes Zielverzeichnis verwendet werden soll, sind in der Datei web.properties im Verzeichnis

prototype\asainlog die Pfade für die Variablen web.server.uddi.catalina.home sowie web.server.uddi.catalina.context.dir entsprechend anzupassen.

Weiterhin ist die Systemvariable CATALINA_HOME mit dem Wert C:\Programme\apache-tomcat-5.5.16-jade zu belegen bzw. mit einem entsprechenden Wert anzupassen, wenn ein anderes Installationsverzeichnis verwendet wird.

Wenn für die Applikation ein anderes Zielverzeichnis als C:\prototype (siehe dazu Abschnitt A.11.1) gewählt wurde, sind in der Datei web.properties die folgenden Variablen entsprechend anzupassen:

- web.server.portal.jboss.home
- web.server.portal.lib
- web.server.portal.liferay.deploy

In der Datei wsig.properties im Verzeichnis prototype\asainlog ist in der Variable wsig.agent_id der Rechnername des Zielrechners zu spezifizieren, auf den das Verzeichnis prototype kopiert wurde.

Die Variable web-subscribe-ws in der Datei build.xml im Verzeichnis prototype\asainlog ist mit dem Namen des Zielrechners zu belegen.

Da das eingesetzte Portal Liferay seine Daten in einer Datenbank ablegt, sind die relevanten Tabellen mit den jeweiligen Daten auf das Zielsystem zu übertragen. Dazu wird die Applikation MySQL Administrator mit dem Benutzer root aufgerufen. In dieser Anwendung werden die beiden SQL-Skripte juddi.sql und lportal.sql aus dem Verzeichnis config des Datenträgers jeweils im Bereich Wiederherstellung geöffnet und in die lokale Datenbank importiert (siehe dazu Abbildung A.11-10 mit Abbildung A.11-13).

Für das Deployment der Portalanwendung sind die folgenden Schritte auszuführen:

- Ausführen des Kommandozeilenbefehls ant web-portal-dist-jar im Verzeichnis prototype\asainlog
- Ausführen des Kommandozeilenbefehls ant deploy im Verzeichnis prototype\ext\ext-ejb
- Ausführen des Kommandozeilenbefehls ant deploy im Verzeichnis prototype\ext\ext-web

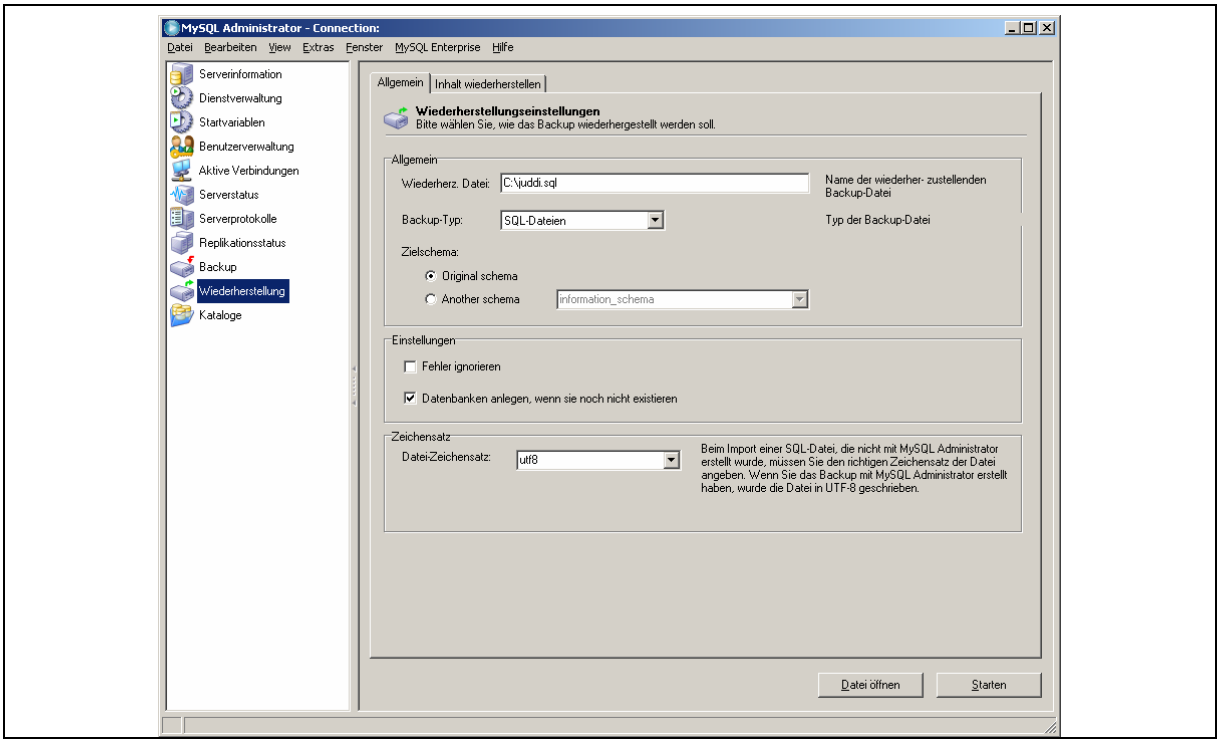

Abbildung A.11-10: Öffnen und Wiederherstellen des SQL-Skripts juddi.sql
Quelle: Screenshot MySQL Administrator

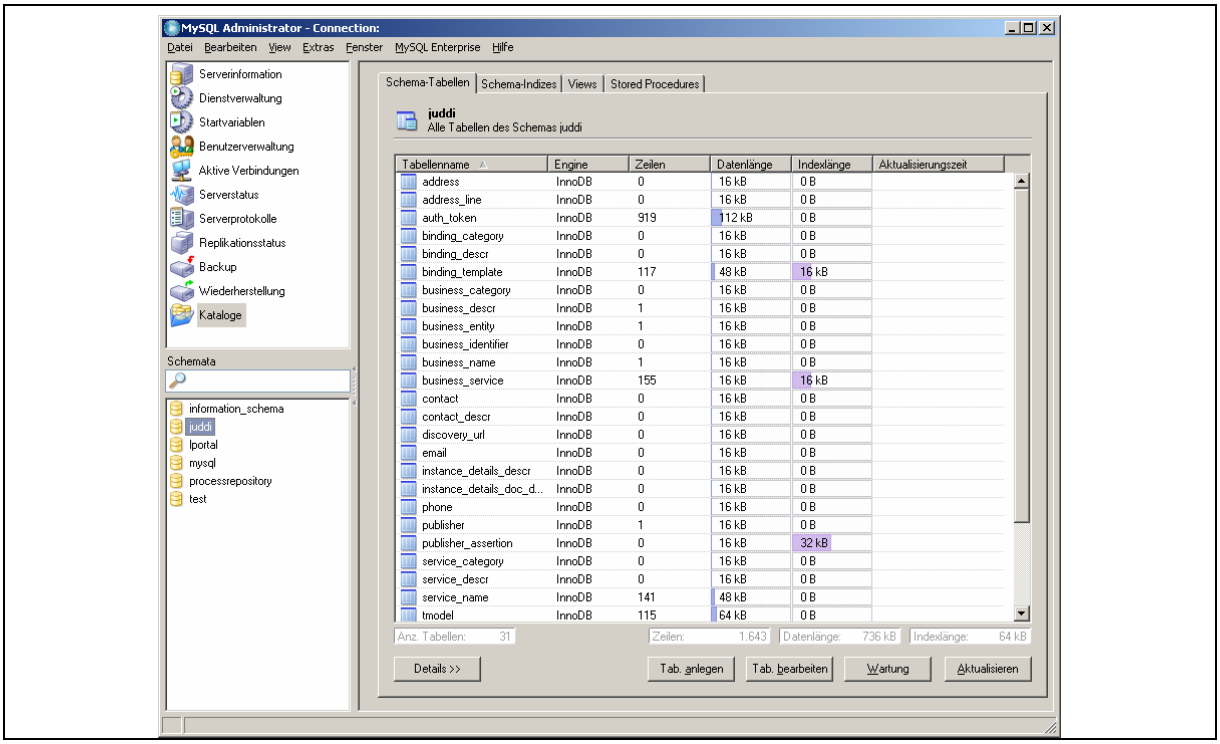

Abbildung A.11-11: Datenbankschema juddi nach seiner Wiederherstellung auf dem Zielrechner
Quelle: Screenshot MySQL Administrator

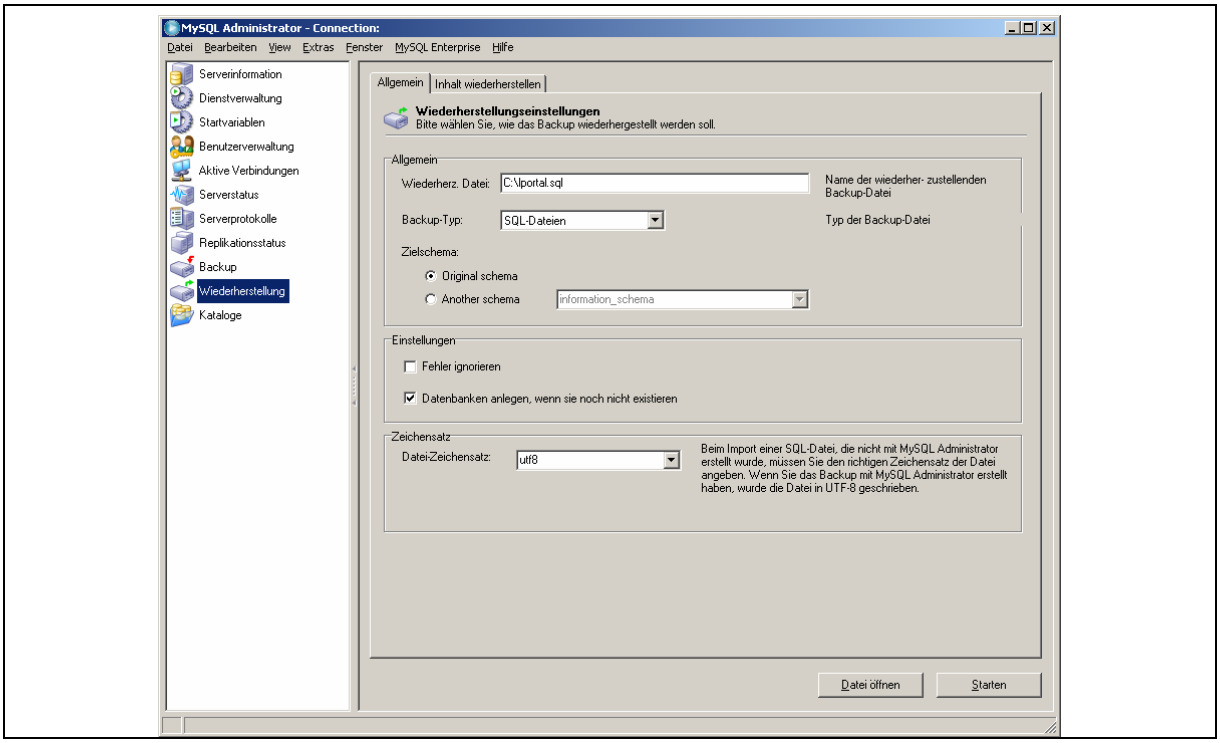

Abbildung A.11-12: Öffnen und Wiederherstellen des SQL-Skripts lportal.sql
Quelle: Screenshot MySQL Administrator

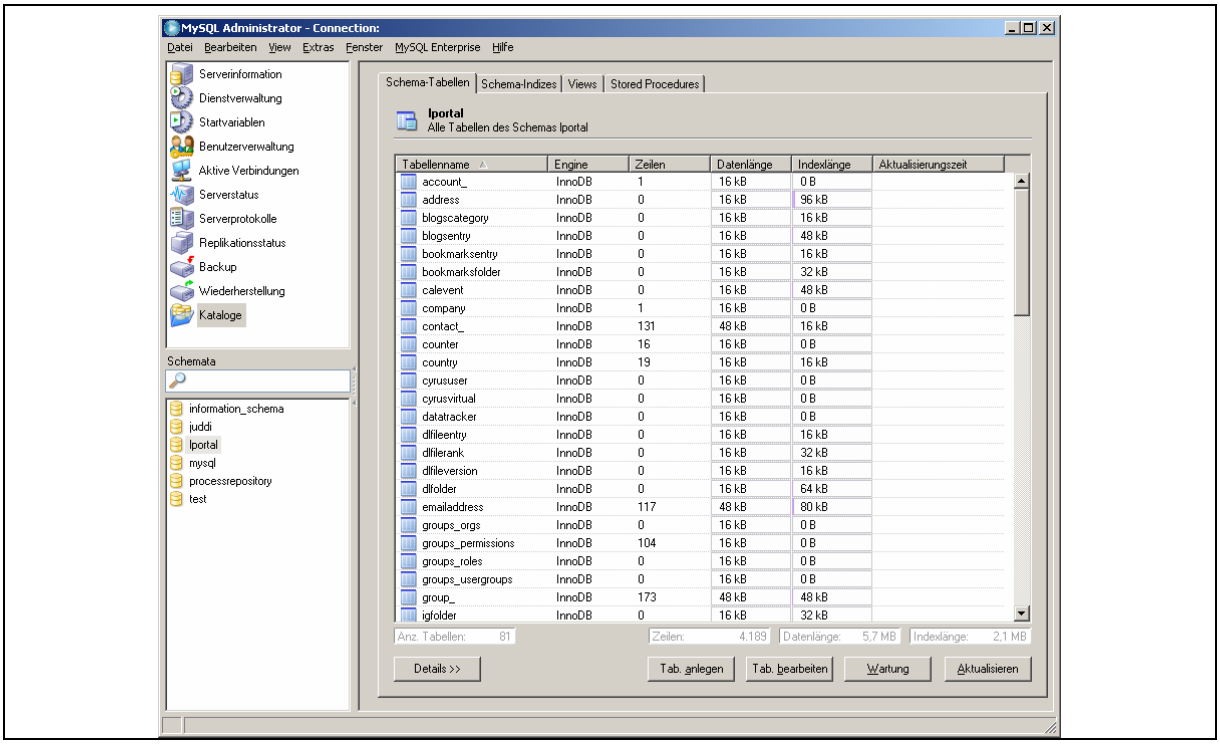

Abbildung A.11-13: Datenbankschema lportal nach seiner Wiederherstellung auf dem Zielrechner
Quelle: Screenshot MySQL Administrator

A.11.5Vorbereitung der Prozesssteuerung

Für die Prozesssteuerung wird in diesem Abschnitt die Anpassung der nach Abschnitt A.11.1 installierten Datenbank beschrieben (für die Darstellung der Installation siehe auch Bergmann/Klinger/Winkler 2006, 94 und 96-96):

Um einen für die Prozesssteuerung erforderlichen Benutzer für die Datenbank anzulegen, ist das Werkzeug MySQL Administrator als Benutzer root aufzurufen (Abbildung A.11-14). Über diese Anwendung wird, wie in Abbildung A.11-15 beschrieben, der Benutzer asainlog mit dem Passwort asainlog angelegt.

Abbildung A.11-14: Anmeldung an der Anwendung MySQL Administrator mit dem Benutzer „root“
Quelle: Screenshot der Applikation MySQL Administrator

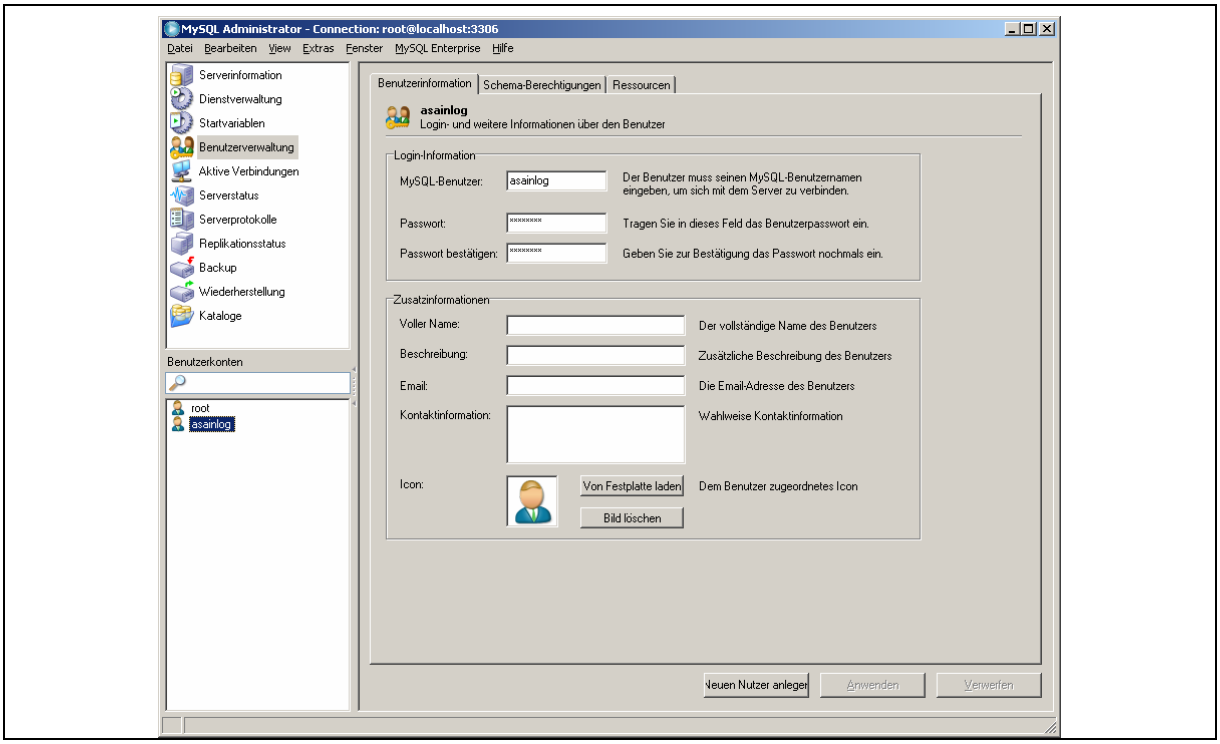

Abbildung A.11-15: Anlegen des Benutzers „asainlog“ mit dem Passwort „asainlog“
Quelle: Screenshot der Applikation MySQL Administrator

Nach dem Starten der Anwendung MySQL Query Browser und der Anmeldung als Benutzer root ist das Datenbankschema processrepository (siehe Abbildung A.11-16) anzulegen. Über die Applikation MySQL Administrator wird unter dem Benutzer root für den Benutzer asainlog die in Abbildung A.11-17 dargestellte Rechtezuweisung vorgenommen.

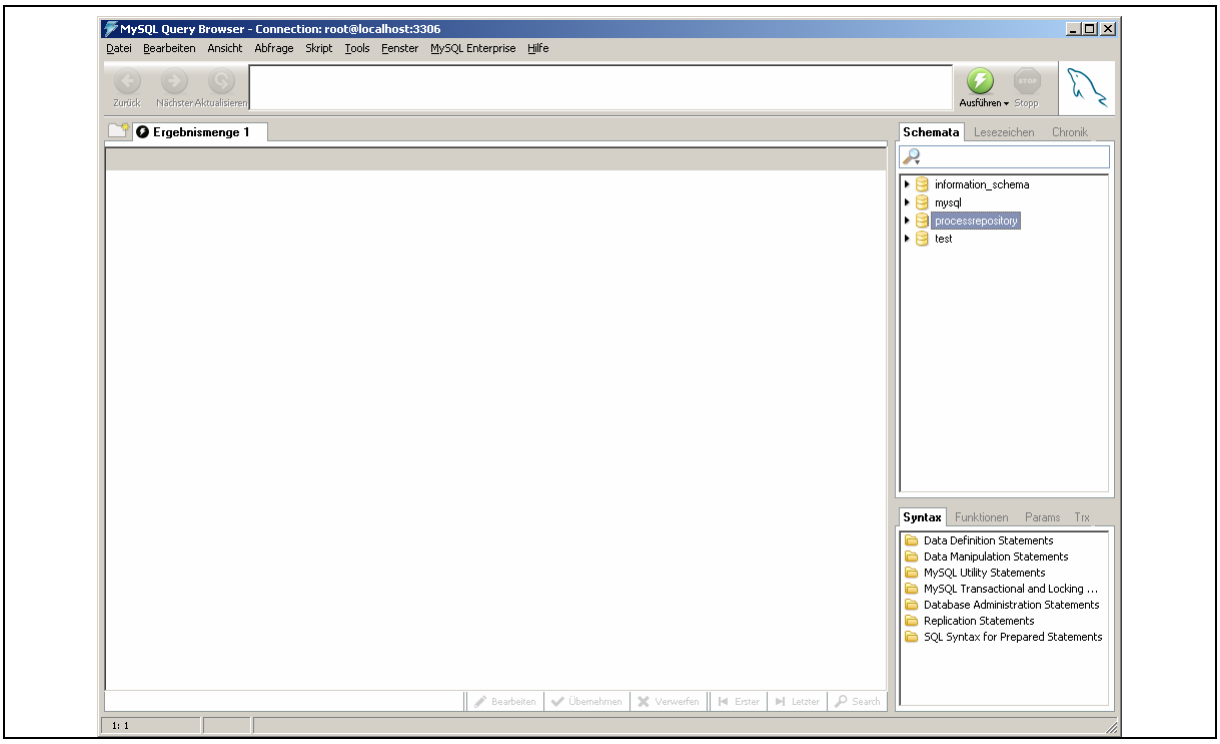

Abbildung A.11-16: Anlegen des Datenbankschemas „processrepository“
Quelle: Screenshot der Applikation MySQL Query Browser

Abschließend wird mit den folgenden Schritten die Information für den modellierten Behandlungsprozess in die Datenbank importiert:

- Ausführen des Kommandozeilenbefehls ant medpath-create-aris-DB im Verzeichnis prototype\asainlog
- Ausführen des Kommandozeilenbefehls ant medpath-import-musterprozess im Verzeichnis prototype\asainlog

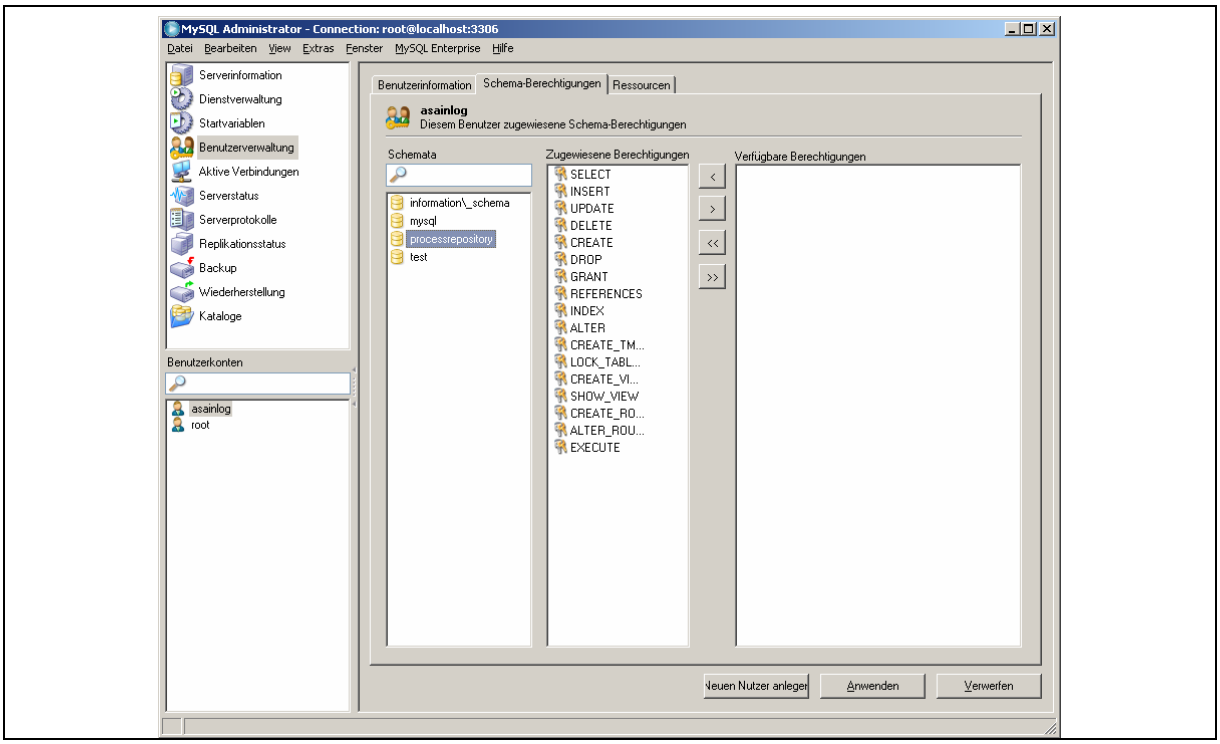

Abbildung A.11-17: Zuweisung von Rechten an den Benutzer „asainlog“
Quelle: Screenshot der Applikation MySQL Administrator

A.11.6 Starten der Desktop-Applikation

Zum Starten der gesamten Anwendung einschließlich seiner Teilapplikationen ist der folgende Befehl auszuführen:

- Aufruf des Kommandozeilenbefehls ant run im Verzeichnis prototype\asainlog

A.11.7 Starten der mobilen Anwendung

Zum Starten der mobilen Anwendung sind die folgenden Schritte (siehe auch Hillebrand 2006, 113) auszuführen, wobei die dargestellte Reihenfolge unbedingt einzuhalten ist:

- Ggf. Starten der gesamten Desktop-Applikation (siehe Abschnitt A.11.6)
- Aufbau einer IP-Netzwerkverbindung zwischen stationärem Rechner und dem PDA (siehe Abschnitt A.11.3)
- Start der mobilen Anwendung über einen Klick auf die Verknüpfung ASAinlogMobile.lnk auf dem PDA

Bei der Bedienung der mobilen Anwendung ist darauf zu achten, dass der an der stationären Anwendung angemeldete Benutzer demjenigen entspricht, der an der mobilen Anwendung angemeldet ist.

A.11.8 Starten der Portalanwendung

Zum Starten des Portal-Servers wird im Verzeichnis ext\servers\jboss-tomcat\bin die Datei run.bat ausgeführt. Sobald dieser Server gestartet ist, kann im Browser das Portal über die Adresse http://localhost:8080 geöffnet werden.

Für die Nutzung der Portalfunktionalität ist die Anmeldung am System erforderlich (siehe Abbildung A.11-18). Dabei ist einer aus den in Tabelle A.11-1 dargestellten Benutzernamen für den Patienten und das zugehörige Passwort auszuwählen. Bei der Wahl des Patienten ist unbedingt darauf zu achten, dass dessen Name mit demjenigen übereinstimmt, dessen eGK aktuell im Kartenleser der Desktop-Applikation eingelegt ist. Damit wird die eKiosk-Funktionalität im Sinne der Telematikinfrastruktur (siehe Abschnitt 6.7.5.1) abgebildet.

Weiterhin ist in diesem Zusammenhang zu betonen, dass für eine Nutzung dieser eKiosk-Anwendung eine SMC vorgesehen ist. Zur Reduktion der Implementierungskomplexität in der vorliegenden Arbeit wird beim Starten der Portalanwendung ein fester HBA aus den in Tabelle A.11-1 aufgeführten Anwendern vorgegeben. Dadurch wird eine eingelegte SMC simuliert. Beim Starten der Portalanwendung ist für ihren fehlerfreien Ablauf zu gewährleisten, dass in den Kartenleser der HBA vom Benutzer obermann eingelegt ist, da dessen HBA als SMC hinterlegt ist. Dementsprechend ist auch eine Anmeldung am medizinischen Informationssystem mit diesem Benutzernamen erforderlich.

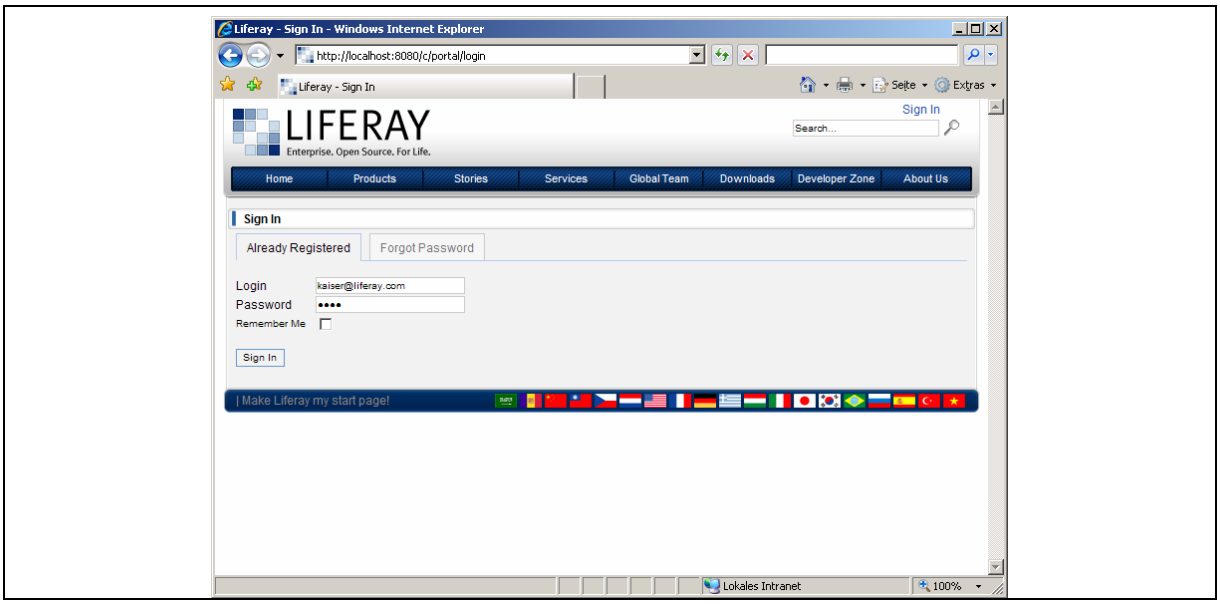

Abbildung A.11-18: Anmeldung am Portal
Quelle: Screenshot Liferay

Nach erfolgreicher Anmeldung am Portal kann eine (beschränkte) Einsichtnahme des Patienten in die bisherige Krankheitsgeschichte (siehe Abschnitt 6.8.6) vorgenommen werden. Weiterhin erhält der Patient die Möglichkeit, seine sportliche Aktivität zu dokumentieren bzw. in einem Überblick zu visualisieren (siehe Abschnitt 6.8.6).

A.11.9 Starten der Prozesssteuerung

Nach dem Start der gesamten Anwendung (siehe Abschnitt A.11.6) ist im Werkzeug SeSAm die folgende Konfiguration vorzunehmen: Im Menü Plugins bzw. im Untermenü FIPA wird der Eintrag Configure DF aufgerufen. Dabei werden die in Abbildung A.11-19 dargestellten Einstellungen vorgenommen. Für <Computername> ist der Name des Zielrechners anzugeben. Abgeschlossen wird die Konfiguration von SeSAm über den Aufruf des Eintrags Configure JADE im Menü Plugins bzw. FIPA. Dort sind die in Abbildung A.11-20 dargestellten Einstellungen vorzunehmen. Dabei ist wieder der Rechnername entsprechend anzugeben. Außerdem kann die in diesem Fenster vorgeschlagene IP-Adresse für den Host übernommen werden. Abgeschlossen wird die Konfiguration durch einen Klick auf den Button Start.

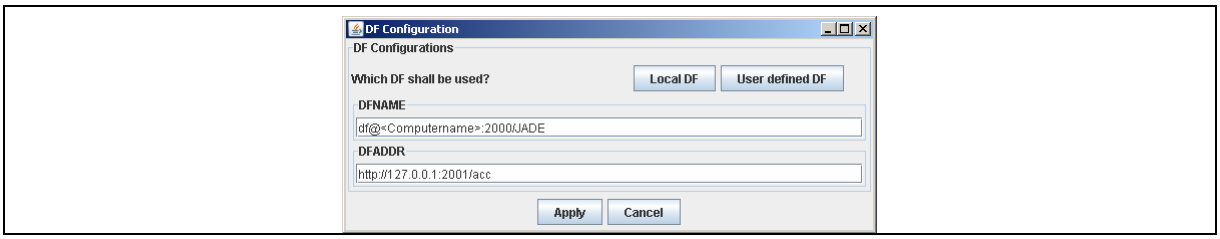

Abbildung A.11-19: Konfiguration des DF im SeSAM-Werkzeug
Quelle: Screenshot SeSAm

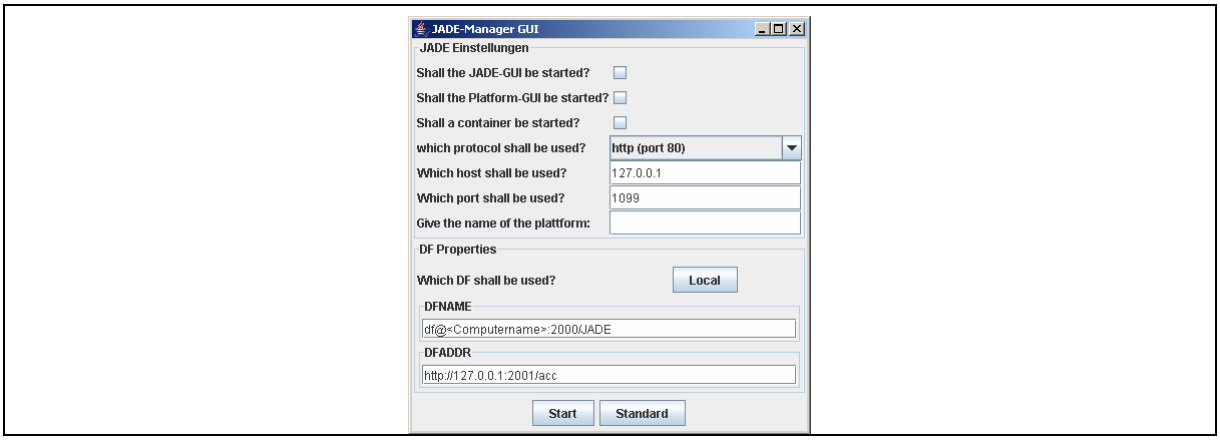

Abbildung A.11-20: Konfiguration von JADE im SeSAM-Werkzeug
Quelle: Screenshot SeSAm

Um die Kommunikation zwischen dem in der vorliegenden Arbeit beschriebenen Prototyp und der Simulationsumgebung SeSAm zu ermöglichen (siehe auch Bergmann/Klinger/Winkler 2006, 99-101), sind in SeSAm (siehe Abbildung A.11-21) die folgenden Schritte vorzunehmen:

- Öffnen der Simulationsumgebung über den Button Start Simulation der Applikation SeSAm:

- Starten der Simulation über den Button Play (siehe Abbildung A.11-21)

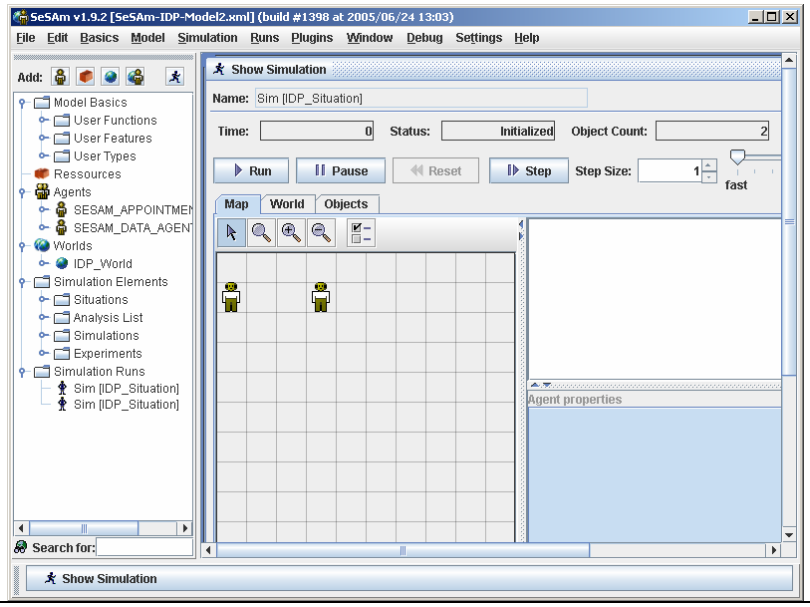

Abbildung A.11-21: SeSAm-Simulationsumgebung für die Interaktion mit der Implementierung
Quelle: Screenshot SeSAm