Entwurf der Architektur für ein Peer-To-Peer basiertes verteiltes Serversystem zur Unterstützung von E-Learning Applikationen

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Lutz Finsterle

geb. in Berlin

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. mult. Paul J. Kühn
1. Mitberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
2. Mitberichter:	Prof. Dr.-Ing. Andreas Kirstädter
Tag der Einreichung:	23.01.2009
Tag der mündlichen Prüfung:	23.07.2009

Institut für Kommunikationsnetze und Rechnersysteme
der Universität Stuttgart
2009

Für meine über alles geliebte Frau Ulrike,
ohne deren Kraft und Unterstützung diese Arbeit nie fertig geworden wäre.

Darüber hinaus danke ich dem Institut für Kommunikationsnetze und Rechnersysteme für die Möglichkeit, die Arbeit durchführen zu dürfen und im Besonderen
Herrn Prof. Dr.-Ing. Dr. h. c. mult. Paul J. Kühn für die Unterstützung in der Endphase.

Abstract

This report describes the architecture, its elements and the application scenarios of E-Learning applications supporting peer-to-peer based server systems. Basic elementary concepts are developed, on how the proposed server system can be rated in terms of its acceptance according to usefulness and especially its capabilities to fulfill the defined requirements, as well as its scaleability to European or even world wide usage. At first an outline of this report is given. Afterwards the two major topics of this work are presented: Easy creation of new as well as reuse of already existing E-Learning materials with most common authoring systems and wordprocessors, and a server system architecture aimed to support a wide variety of E-Learning applications conforming to the constraints driven by pedagogical, social and business criteria.

1 Structure of the report

Chapter 1 - Introduction

Emphasising todays and future relevance of media enhanced learning, the relevant problems and issues are pointed out that do arise while introducing these media into todays "learning environments" like universities, schools and other institutional and non institutional environments, corporations and SME's[1].

Chapter 2 - State of the art

When talking about "media enhanced learning" there are technical as well as pedagogical and psychological aspetcs as well as others, driven by "external" processes, that must be explained to fully understand why certain problems arise therefrom. The basic technical knowledge and major driving issues of pedagogical and social influence are provided here. Definitions, terms and "State of the Art" technological and pedagogical background is referenced or, if literature is not exact or clear enough, explained in the necessary depth to understand the major topics of this work. Having introduced the basics, the following chapter explains the boundary conditions and usage scenarios this work is based on.

[1] Small to Medium sized Enterprises

Chapter 3 - Constraints and usage scenarios
Due to the complexity of the problem, exact definitions of the basic boundary conditions as well as of the considered usage scenarios must be given. This sets the framework for all further reasoning and the development of the herein described architecture. The chapter focuses mainly on the user demands, their mobility[2] and technical as well as common social and economic issues relevant to the design and architectural decisions made in the present work.

As can be derived from those constraints and scenarios, not only the technical issues of the platform but also the ease of, and support for, E-Learning material creation drives the acceptance of such kind of a system. To overcome this drawback, present in many of todays E-Learning systems, the next chapter describes a widely applicable framework for creating E-Learning content.

Chapter 4 - Creation of structured E-Learning material
Also the presentation of learning material employing modern media and presentation systems is still evolving, there is quite a volume of material already existing in one or the other electronic form. If the newly introduced E-Learning system is not or only partially capable of importing these materials, it is very cumbersome to migrate to this system. Furthermore, most of these materials lack the structure and semantic information needed to achieve a reasonable reuse and exchange rate. In this chapter a novel transformation system is proposed as well as its integration in "ordinary" or well-known word processing tools, that enables the authors to import their already existing materials and, with only a small amount of work to be done, transform them in full fledged structured and semantically annotated E-Learning modules.

Now, having those materials as well as applications for teaching available, there must exist the ability to access, reuse and exchange them easily between other teachers, institutes or experts. Surely, that can be done using techniques like WWW or similar communication structures, but this is, as shown in Chapter 2, not seamlessly possible. To overcome arising problems, a system is proposed in the following chapter that aims at seamless use and ease of setup. The system is featuring all capabilities needed to fulfill the demands of the E-Learning enhanced world of the future, as the next chapter will show.

Chapter 5 - Architecture of the proposed server system
The proposed architecture is an open and modular service-based architecture that fulfills all the needs described in the previous sections. Main part is a Micro-Kernel which

[2] Mobility, in this work, is understood as working at different places and, not necessarily while traveling

serves as the central service container and a set of necessary basic services to support the major tasks to be performed by the proposed system. These are: A nearly unconstrained transport and communication layer, a user-management , -profile and -preferences management, and the data-distribution layer based on an intelligent and agile filesystem interface. Some special attention has also been given to a distributed secure user authorization mechanism and anonymous authentication. The main issues here do arise in various areas: One is the handling of anonymous certificates to provide a reasonable amount of "private sphere" and security to the user while still allowing for accounting and billing of the used services. Secondly, related to this, a need arises for secure handling of login and certificates in case of "mobile" users. So, remote code verification with and, more often, without hardware supported "secure memory"[3] comes into the focus. The third issue herein comes from the necessity of reasonable secure billing of cached material, which also has to be thoroughly looked at.

A proof of concept for the server-system is given in the following chapter by description of a partial implementation of the described main architectural and conceptual components.

Chapter 6 - The realized prototype
The prototype system described is based on three student projects, commonly available peer-to-peer frameworks and own developed software based on ideas springing off the projects CANDLE and ITO. As such it is meant as a proof of concept for a variety of small concepts described in this thesis. Basic hints on how to prove the acceptability of the proposed system in terms of bandwidth usage, a measurement and simulation environment has been realized which is also described in this chapter.

Chapter 7 - Conclusions and Outlook
The herein described server system has shown that the high demands put on modern E-Learning environments can be fulfilled with respect to almost all areas.

2 Supporting reuse and exchange of content with distributed access to applications in media enhanced learning environments by example
Considering the issues that have been noted above, a short exemplary introduction to the overall consistent concept of the present work is given by example. Assuming a medium sized institution that wants to shift from their, until now practiced, approach in teaching, which is quite classical, to a modern approach. This shift is driven by changes in the clients demands on how they need to, or at least want to, be educated. A more

[3] Recently, the notion of hardware secured memory areas becomes widely available. See [57].

flexible way of accessing the information and knowledge is aimed, by keeping the communication and social interaction between the learners on the same level. Furthermore the support of ongoing learning to keep their knowledge up to date is demanded for a period of time after having finished the course.

So, a couple of problems arise to the team of the considered institution. The upcoming issues are:

- Prepare material for the new courses, possibly having some volume already prepared
- Find a pedagogical correct way or concept for presenting and deploying such a course employing media enhanced support technologies
- Set up the necessary infrastructure for those media enhanced support technologies, having probably a Web-Server already, but knowing that just downloadable materials are insufficient
- Deploy course material into infrastructure, adding metadata and so on, in an embedded or additional step with even new overhead
- Present and/or give link and access to materials to the learners, in a way that fits the pedagogical concept choosen
- Find a way how to move well-known other institutionalized teaching standards, like workgroups, seminars, laboratory studies and so on to the structures given by means of media enhanced learning

All these steps do put an additional load on the teacher. So the herein described plattform tries to address all those issues in an open way by nontheless presenting a closed loop of tools to the teacher or author. He or she will find a simple to set up environment which allows for easy creation of modularized materials and maintaining their well-known authoring tool with some new extensions. Further features are: easy deployment, further reuse and exchange of those newly prepared course modules, simple integration into already existing infrastructure for presenting those materials and, last but not least, the ability of drawing monetary profits from his or her efforts.

Kurzfassung

Die Vermittlung von Wissen ist in ihrem Wesen ständigen Veränderungen unterworfen. Heute geht die Wissenschaft von der Auffassung aus, dass die Vermittlung von Wissen und Fertigkeiten am besten durch das Konzept des sogenannten „Blended Learning“ [4] zu erreichen ist. Dieses Konzept vereinigt die Konzepte des „personenbezogenen“ [5] Lernens mit den Konzepten des E-Learning.

Parallel zu der Präsenzlehre an Universitäten, Schulen und ähnlichen Institutionen, das heißt der Wissensvermittlung über Vorlesungen, Übungen und Praktika in deren unterschiedlichsten Ausprägungen, hat sich das Wesen der existierenden Informationsquellen über die Zeit hinweg wesentlich verändert. Hierbei spielten im besonderen die Weiterentwicklung der elektronischen Dokumentenverarbeitung und die sich entwickelnde Informationstechnologie in Form der Computernetze eine wesentliche Rolle. Mit der Verfügbarkeit des Internets in Universitäten, Fachhochschulen und anderen wissenschaftlichen Einrichtungen sowie der Einführung des WWW[6] durch Wissenschaftler am CERN[7] wurde es möglich, beliebige Inhalte mit textuellen, grafischen und interaktiven Komponenten weltweit verfügbar zu machen. Sehr bald wurde das WWW zu einer der wichtigsten Quellen von Informationen für die Wissenschaftsgemeinde.

Seit 1997 fördern sowohl die Europäische Kommission als auch die Bundesrepublik Deutschland Forschungsprojekte im Bereich der Lehre mit modernen Medien. Diese Förderprogramme wie auch die sich schnell verbreitende Auffassung, dass die kommende Lehr- und Lernlandschaft zwangsläufig von E-Learning-Konzepten geprägt sein wird, führten zu verstärkten Aktivitäten in diesem Bereich. Hierbei lag der Fokus der Aktivitäten auf den verschiedensten Ebenen. Mit der Zeit stellte sich heraus, dass der

[4] „Blended Learning“ beschreibt eine Mischung aus konventionellen Lernformen und reinem Online-Training

[5] „Personenbezogene Lernformen“ meint in diesem Kontext alle Lernformen, welche ohne elektronische Kommunikationsmedien auskommen

[6] World Wide Web

[7] Die erste Version des HTML-Standards sowie den dazugehörigen Server- und Anwendungskomponenten wurde 1989 von Tim Berners-Lee am CERN erdacht und vorgestellt

ursprüngliche Ansatz der ”reinen” Bereitstellung von Lern- und Lehrmaterialien, Programmen und Informationssammlungen zusammen mit dem Paradigma der vollständig virtuellen Lehre, verglichen mit dem bisher praktizierten Paradigma der anwesenheitsorientierten Lehre, Nachteile und schlechtere Ergebnisse hinsichtlich des Erreichens der Lernziele aufwies. Die Erfahrungen zeigten, dass es außer dem Angebot von Inhalten und Programmen dringend notwendig ist, diese Inhalte durch Präsenzveranstaltungen zu begleiten, um eine Steigerung der Qualität der Lehre zu erreichen. Dieses Resultat beruht auf der aus den begleitenden Präsenzveranstaltungen entstehenden stärkeren Bindung an die Inhalte beziehungsweise an die Veranstaltung selbst. Das hieraus sich ergebene Paradigma wird heute im Allgemeinen als ”Blended Learning” bezeichnet.

Die Aktivitäten in den Bereichen der Inhaltserstellung und Verwaltung sind durch diese Paradigmenverschiebung jedoch nahezu unberührt und stellen sich wie im Folgenden beschrieben dar.

Durch den beständig schneller werdenden Wandel des Wissens in unserer Zeit sowie dem Wandel in der Lehrlandschaft steigen die Anforderungen an die Qualität und damit die Aufwände zur Erstellung und Pflege der Lernmaterialien. Dies führt zu zunehmenden Arbeitsaufwänden bei den mit diesen Aufgaben betrauten Personen und Institutionen.

Besonders im Bereich des E-Learning, das heißt für das Lehren und Lernen mit modernen Medien, ist der Aufwand für die Erstellung der Materialien wesentlich größer als bei der Erstellung von Unterrichtsmaterialien für die “älteren“, herkömmlichen Lernformen.

Als E-Learning Inhalte werden heute alle Inhalte angesehen, die in beliebiger Form elektronisch abrufbar sind und in einer Form dem Nutzer präsentiert werden können, die den pädagogischen Anforderungen, welche im Allgemeinen an Lehrinhalte gestellt werden, gerecht werden. Hierzu gehören zum Einen die entsprechende modulare und annotierte Aufbereitung der Inhalte als auch eine entsprechende Zugriffs- und Präsentationsumgebung. Gerade aus der Notwendigkeit, die Inhalte zu modularisieren und zu beschreiben, erwachsen neue Arbeitsaufwände bei den Autoren. Ohne diese Aufwände ist es jedoch nicht möglich, die Zugriffs- und Präsentationsumgebungen, wie zum Beispiel Lernportale, mit den pädagogisch notwendigen Eigenschaften der Lernerführung und Personalisierbarkeit, auszustatten.

Es ergeben sich daneben auch Vorteile aus der Modularisierung und Annotation der Materialien. Werden hierbei bestimmte Randbedingungen eingehalten, so können die erstellten Materialien wiederverwendet und mit anderen Anbietern ausgetauscht werden. Die sich damit ergebende Nachhaltigkeit der Nutzung der erstellten Inhalte und der damit sich wesentlich reduzierende Aufwand bei der Erstellung neuer Kurse ist evident

und überzeugend.

Der Inhalt der vorliegenden Arbeit dient der Entwicklung und Basisbewertung einer Architektur sowie den notwendigen Hilfsmitteln für Inhaltsersteller, Lehrer und Lernende auf Basis der in den letzten Absätzen gemachten Vorbemerkungen sowie den daraus sich ergebenden Randbedingungen.

Ziel ist es, ein Serversystem, das die verschiedensten E-Learning-Anwendungen unterstützt und gerade den Austausch und die Wiederverwendung beziehungsweise Einbettung von bereits vorhandenen Materialien ermöglicht, zu entwickeln. Hierbei steht besonders auch die Abrechenbarkeit der Inhalte im Vordergrund. Ohne diese ist eine zukünftige Lehrlandschaft nur schwer vorstellbar, da die Notwendigkeit zur Refinanzierung der entstandenen Aufwände nicht nur für kommerzielle Anbieter wichtig ist. Die Entwicklung der Universitätslandschaft zeigt, dass der Verkauf der Inhalte, auch zur Finanzierung der Hochschulen, ein wesentlicher Aspekt sein wird. Das Serversystem kann hier zwar weitgehend unabhängig von der Materialerstellung betrachtet werden, aber die Materialerstellung ist auf Grund des geschilderten, hohen Aufwandes nicht vernachlässigbar. Deshalb sollen ebenfalls die notwendige Strukturierung, die Erstellung sowie Umwandlung von Lehrmaterialien in strukturierte und metadatenbehaftete Lehr- bzw. Lernmodule und, wie im Folgenden ausgeführt wird, die verteilte Nutzung, das heißt das Suchen und der Zugriff auf diese behandelt werden.

Die erstellten Materialien müssen nun, um eine nachhaltige Nutzung durch Austausch und Wiederverwendung sicherzustellen, auch auffindbar und zugreifbar bereitgestellt werden. Es wird ein System benötigt, welches es den Institutionen unter den geschilderten Randbedingungen ermöglicht, die von ihnen erstellten Inhalte anderen in abrechenbarer Form zur Verfügung zu stellen und selbst Materialien der anderen Institutionen zu nutzen, ohne dass dadurch erneut ein zusätzlicher Aufwand entsteht. In der vorliegenden Arbeit wird hierzu ein Peer-to-Peer basiertes Serversystem entworfen, welches einen ganzheitlichen Ansatz zur Verteilung und Wiederwendung beliebiger Lehrangebote und Lehranwendungen verfolgt. Das System bietet durch den unterliegenden Peer-to-Peer-Kommunikationsmechanismus die einfache Möglichkeit einer umfeldunabhängigen Installation und Teilnahme an dem Lehrangebotsaustauschnetz unter Berücksichtigung von Anforderungen wie Abrechenbarkeit und Verfügbarkeit sowie Integration in das bestehende Umfeld der Institution.

Wesentliche Aspekte der Arbeit sind somit zum einen die Form der Inhalte und deren Erstellung, die Unterstützung von Austausch und der Wiederverwendung von Lehrmaterialien, sowie eine Infrastruktur, welche es ermöglicht, beliebige Applikationen mit ihnen

zu verknüpfen.

Inhaltsverzeichnis

Abstract i
Kurzfassung v
Inhalt ix
Abbildungen xii
Tabellen xv
Abkürzungen und Zeichen xvii

1 Einleitung **1**
1.1 E-Learning als neues Medium in der Lehre 1
1.2 Motivation 5
1.3 Zielsetzung und Abgrenzung 6
1.4 Gegenstand der Dissertation 7
1.5 Struktur der Arbeit 7

2 Grundlagen **9**
2.1 Technische Grundlagen des E-Learning 10
2.1.1 Modularisierung von Lehr- und Lernmaterialien 12
2.1.2 Content Management 19
2.1.3 Spezifikation von Metadaten für Lehrmodule 20
2.1.4 Suchmechanismen für Lehrinhalte 21
2.2 Beschreibungen von Modulkompositionen 21
2.3 Netz- und systemtechnische Grundlagen 22
2.3.1 Das Internet und seine wesentlichen Protokolle 23
2.3.2 Sicherheitsmechanismen im Internet 25
2.3.3 Verteilte Systeme und deren Kommunikationsmechanismen 26
2.3.4 Authentication, Authorization und Accounting (AAA) 31
2.4 Das World Wide Web (WWW) und seine Entwicklungen 35
2.4.1 WWW 35
2.4.2 Das Semantic Web 35

2.4.3 Ontologie-getriebene Suche und Strukturierung 36
2.5 Grundlagen von Peer-to-Peer-Systemen 37
2.5.1 Paradigmen von Peer-to-Peer-Konzepten 37
2.5.2 Entwicklung der Peer-to-Peer-Systeme 38
2.5.3 Grundmechanismen heutiger Peer-to-Peer Systeme 38
2.5.4 P2P-Systeme und ihre Eigenschaften 40
2.6 Abgeleitetes Anforderungsprofil an ein Server-System 44

3 Grundkonzepte einer verteilten Server-Architektur für E-Learning 47
3.1 Spezifikation der Systemnutzer 48
3.2 Szenarien der Nutzung 49
3.3 Äußere Rahmenbedingungen 51
3.4 Wirtschaftliche Anforderungen 51
3.5 Funktionale Anforderungen 52
3.6 Abgeleitete Grundkonzepte 53

4 Struktur und Erstellung von E-Learning-Materialien 55
4.1 Lehrmodulstrukturen und herkömmliche Autorensysteme 55
4.2 Formatvorlagen zur automatisierten Transformation 56
4.3 Erweiterungen im Autorensystem zur Nutzerunterstützung 59
4.4 Einbettung von Metadaten 59
4.5 Transformation in Module und Metadaten 62
4.6 Das Ausgabeformat 70
4.6.1 Unified Resource Identifier 70
4.6.2 Schemadefinition des Zwischenformates 72
4.6.3 Darstellung von Inhalten im Zwischenformat 73

5 Die Architektur des Server-Systems 75
5.1 Basisstruktur des Kernels 75
5.2 Hierarchische Struktur und Namensräume des DLE 77
5.3 Kommunikationsmechanismen für die DLE-Dienste 79
5.3.1 Die Transportschicht 80
5.3.2 Der DLE-Socket-Service 81
5.4 DLE-Dienste 82
5.4.1 Nutzer-Management 82
5.4.2 Login (Authentication) und Pseudonymverwaltung 83
5.4.3 Nutzerprofil-Verwaltung 83

5.4.4 Suche 85
5.4.5 Caching 86
5.4.6 Virtuelles File-System (VFS) 86
5.5 Die Datenablagestruktur des DLE 95

6 Simulation und prototypische Realisierung zur Abschätzung der Systemeigenschaften 99
6.1 Modellierung des Systems und seiner Teileigenschaften 100
6.1.1 Modellierung der SRDI DHT-Implementierung 101
6.1.2 Übermittlung von Login-Daten, Pseudonym-Daten und Login-Modulen 101
6.1.3 Übertragung von Personalisierungsdaten 102
6.1.4 Übermittlung von Caching-Daten und -Modulen 102
6.1.5 Übermittlung von Accounting-Daten 103
6.1.6 Datenaufwand des Virtuellen Filesystems 103
6.2 Einordnung der betrachteten Teilsysteme 103
6.2.1 Bewertungskriterien für das Server-System 104
6.2.2 Peer-to-Peer-Nachrichtenaustausch 106
6.2.3 Der Systemkern (Micro-Kernel) 107
6.2.4 Das AAA-Teilsystem 107
6.2.5 Realisierung der Transportschicht 107
6.2.6 Realisierung der Filesysteme 108
6.2.7 Realisierung des Such-Dienstes 109
6.3 Durchführung der Simulation und Messung 109
6.3.1 Simulation des Peer-to-Peer Netzes 109
6.3.2 Messungen am Prototyp 113
6.4 Ergebnisse aus Untersuchungen und Abschätzungen 120
6.4.1 Simulierte Teilbereiche 120
6.4.2 Realisierte Teilbereiche 124
6.4.3 Abgeschätzte Teilbereiche 125
6.4.4 Ergebnisse aus den Betrachtungen 129
6.4.5 Aufwandsabschätzungen 131

7 Zusammenfassung und Ausblick 133

Literaturverzeichnis 141

Index **149**

Anhänge

A Simulations- und Messwerkzeuge **151**
A.1 Event Basierte Simulationsbibliothek 151
A.1.1 Komponenten der Simulationsbibliothek 152
A.1.2 Die Konfigurationsdatei 153
A.1.3 Lebensphasen der Komponenten 156
A.2 Beispiel eines Modells 157
A.2.1 Aufbau des Modells 157
A.2.2 Implementierung des Modells 158
A.2.3 Konfiguration des Simulationsmodelles 158
A.3 Einflussmöglichkeiten auf die Simulation 158
A.4 Messwerkzeug für Prototypen 159

B Kurse und Kursstrukturen **169**
B.1 Aufbereitung der Lernmaterialien in CANDLE 171
B.2 Aufbereitung der Lernmaterialien in ITO 172
B.3 Berechnung der Inhalts- und Metadatengrößen 172

C Anwendungen und deren Anforderungen **177**

D Ergänzungen **179**
D.1 Formatdefinition des Zwischenformates 179
D.2 Vollständige Übersicht über die Basis-Datenstrukturen des DLE 180
D.3 Klassenbaum der Simulationsbibliothek 184
D.4 Strukturdefinition der Konfigurationsdatei der Simulationsbibliothek 185

Abbildungsverzeichnis

2.1 Vom E-Learning-Hype zur Nachhaltigkeit 11
2.2 Entwicklung des E-Learning-Anteils in den Hochschulen 13
2.3 Referenzarchitektur eines durchgängigen E-Learning-Systems 14
2.4 Basis-Struktur der Lehrmodule 16
2.5 Zusammensetzung von Lehrmodulen 18
2.6 Ein einfaches Netz 23
2.7 Internet-Protokoll-Schichtung 24
2.8 Einsatz von Firewalls im Internet 25
2.9 Grundmechanismus von entfernten Funktionsaufrufen 27

3.1 Nutzerrollen und Systemaufgaben 49

4.1 Formatvorlagen zum Erstellen von Lehrmodulen 57
4.2 OpenOffice.org als Autorensystem 60
4.3 Werkzeugkette für den Autor 61
4.4 Metadaten Annotations Dialog 63
4.5 Der Transformationsprozess 64
4.6 Ablaufdiagramm des Transformationsprozesses 65
4.7 Java-Implementierung des Handler Interface 68
4.8 Ablaufdiagramm der Handler-Nutzung 69
4.9 Struktur des Ausgabeformates 71
4.10 In HTML transformiertes Lehrmodul 74

5.1 Übersicht über das DLE-System 76
5.2 Basisdienste des Mirco-Kernels 76
5.3 Die Gruppenstruktur des DLE 77
5.4 Aufbau der Adresssätze 78
5.5 Abstraktion des DLE-Systems 78
5.6 Gruppenstruktur des Prototypen 79

5.7 Der Transportmechanismus 81
5.8 Schematische Darstellung der Profildatenablage 84
5.9 Zugriff auf lokalen Cache 88
5.10 Zugriff auf verteilte Caches 89
5.11 Overlay-Dateibaumstruktur eines Servers 93
5.12 Interne Datenstruktur des DLE-Systems(Lehrmodule) 96
5.13 Interne Datenstruktur des DLE-Systems(Lehrmodule) 97
5.14 Interne Datenstruktur des DLE-Systems(Lehrmodule) 98

6.1 Verkehrsaufkommenrelevante Teile des Systems 100
6.2 Simulationsmodell 110
6.3 Die Logik der SRDI-Implementierung (Task zum bearbeiten eingehender Packete) 112
6.4 Die Logik der SRDI-Implementierung (Löschen veralteter Nutzdaten im Knoten) 113
6.5 Die Logik der SRDI-Implementierung (Verarbeiten von eingehenden Paketen) 114
6.6 Die Logik der SRDI-Implementierung (Verteilen von Ändernungen) 115
6.7 Die Logik der SRDI-Implementierung (Verschiedene Funktionen der Paketverarbeitung) 116
6.8 Die Logik der SRDI-Implementierung(Replizierung der Nutzdaten im Knoten) 117
6.9 Knotenparametrisierung der SRDI-Simulation 118
6.10 Steuerdatei der SRDI-Simulation 119
6.11 Konfiguration der Mess-Bibliothek 120
6.12 Bandbreite am Eingang des Netzes 121
6.13 Durchschnittliche Bandbreite an den Netzknoten 122
6.14 Datenrate 200 Knoten, 1000 Einträge, Line-Rate 260kB/s 123
6.15 Datenrate 200 Knoten, 2000 Einträge, Line-Rate 260kB/s 124
6.16 Datenrate 400 Knoten, 2000 Einträge, Line-Rate 260kB/s 125

A.1 Die Komponenten-Konfiguration 155
A.2 Modell der Beispielsimulation 157
A.3 Basisklasse SimModel 159
A.4 MySimModel 161
A.5 MySimModel cont. 162
A.6 Ein einfacher Knoten für das Simulationsmodell 163

A.7 Konfiguration MySimModel 164
A.8 Konfiguration MySimModel 165
A.9 Startup Simulation Sequence Diagramm 166
A.10 Interaktive Lehranwendung 167
A.11 Abfangen der send() Methode 167

B.1 Ursprüngliches Vorlesungsmaterial 173
B.2 Aufbereitetes Vorlesungsmaterial 174
B.3 Exemplarischer Metadatenauszug 175
B.4 Das Candle Authoring Tool (CAT) 176

D.1 Interne Datenstruktur des DLE-Systems 181
D.2 Interne Datenstruktur des DLE-Systems 182
D.3 Interne Datenstruktur des DLE-Systems 183
D.4 Pakete des Simulations-Klassenbaumes 184
D.5 SimConfig DTD (Teil 1) 186
D.6 SimConfig DTD (Teil 2) 187

Tabellenverzeichnis

2.1 Basisprotokolle des Internets 24

4.1 Element Handler für das Zwischenformat und OpenOffice 67

6.1 Größenabschätzung der zu übermittelnden Daten 102
6.2 Übersicht über die Modulgrößen verschiedener Medientypen 127
6.3 Institute an deutschen Hochschulen nach [44] 129

B.1 Kurse und deren Autoren 170
B.2 Kurs- und Metadaten-Größen 171

C.1 Anforderungen der Anwendungen 178

Abkürzungsverzeichnis

A4C	AAA + Auditing + Charging
AA	Active Application
AAA	Authentication, Authorization and Accounting
ACL	Access Control List
AFS	Andrew File System
CBT	Computer Based Training
CIFS	Common Internet Filesystem
CMS	Content Management System
CSS	Cascading Stylesheets
DARPA	US Defence Advanced Research Project Agency
DC	Dublin Core Standard for Learning Module Metadata
DHT	Distributed Hash Table
DLE	Distributed Learing Environment
DOM	Document Object Model
HTML	Hyper Text Meta Language
HTTP	Hyper Text Transfer Protocol
JXTA	Java/C++ basiertes Peer-to-Peer Basis-System
LMS	Learning Management System
NFS	Network File System
P2P	Peer-to-Peer
RDF	Resource Description Framework
RFC	Request for Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SMB	Server Message Block
SOAP	Simple Object Access Protocol
SRDI	Shared Resources Distributed Index
SSL	Secure Session Layer

STLP ..... Secure Transport Layer Protocol
SVG ..... Scaleable Vector Graphics
TCP ..... Transfer Control Protocol
TLS ..... Transport Layer Security
UDP ..... Universal Data Packet
UML ..... Unified Modelling Language
VFS ..... Virtual File System
VFSI ..... Virtual File System Interface
W3C ..... World Wide Web Consortium
WBT ..... Web Based Trainings
WEB-DAV ..... Web-based Distributed Authoring and Versioning
WSDL ..... Web Service Description Language
WWW ..... World Wide Web
XML ..... eXtensible Markup Language
XML-DOM ..... XML Document Object Model
xmlrpc ..... XML basiertes RPC Protokoll

Kapitel 1

Einleitung

1.1 E-Learning als neues Medium in der Lehre

Die Entwicklung der letzten Jahre im Bereich des E-Learnings, im Besonderen im Hinblick auf die nachhaltige Unterstützung und Einführung im universitären Lehrbetrieb, hat gezeigt, dass die Komplexität dieser Aufgabe anfänglich unterschätzt wurde. Hierbei sind die wesentlichen Faktoren, welche eine breite nachhaltige Akzeptanz der in Forschungsprojekten und Firmen entwickelten Systeme deutlich eingeschränkt haben, zum Einen die mangelnde Flexibilität der Systeme im Bereich der Inhaltsdatenerstellung und -konversion und zum Anderen die durchweg zentralisierte Administration und Installation der existierenden Systeme.

Ein weiterer wichtiger Punkt ist das Eingehen auf die aus der Pädagogik aufkommenden Anforderungen und den gesellschaftlichen Randbedingungen, denen wir heute unterliegen. Zum Einen ist hier die Forderung nach der Individualisierung der Lernangebote zu nennen, zum Anderen das „Learn-Everytime-Everywhere“-Paradigma, das es zumindest auch um „Teach-Everytime-Everywhere“ zu erweitern gilt. Es stellt sich hier darüberhinaus die Frage, ob es auch ein „Authoring-Everytime-Everywhere“ geben muss, oder ob diese Anforderung eher einzuschränken ist.

Betrachtet man die Problematik der Inhaltserstellung im Allgemeinen, so stellt diese,

auch in der Form wie sie heute größtenteils noch betrieben wird, bereits einen großen Aufwand für die Lehrenden und die Autoren von Lehrmaterialien dar. Die sich, gerade im Bereich der Informationstechnologie, sehr schnell wandelnde Wissenslandschaft erfordert einen ständigen Pflege- und Anpassungsaufwand der Lehrmaterialien. Mit dem Einzug neuer Medien[1] in die Lehrlandschaft hat sich der anfallende Aufwand für diese Arbeiten noch weiter erhöht.

Durch den Einsatz von E-Learning Systemen steigt dieser Aufwand nun noch einmal. Denn, für einen erfolgreichen Einsatz eines solchen Systems ist es nicht ausreichend, die existierenden Materialien in elektronischer Form bereitzustellen. Vielmehr müssen, soll aus dem Einsatz dieser Technologien nicht ein Verlust für die etablierte Bildungslandschaft sowie eine Verschlechterung der Lehre resultieren, wesentliche soziale und pädagogische Randbedingungen beachtet und die Materialien an die veränderten oder neuen Randbedingungen angepasst werden. Darüber hinaus müssen die Materialien mit zusätzlichen Anmerkungen[2] versehen werden um deren Pflege und erweiterte Verwendbarkeit, wie zum Beispiel den Austausch oder die Wiederverwendung der erstellten Materialien in einem anderen Kontext, zu vereinfachen oder erst zu ermöglichen.

Die oben angesprochenen sozialen Randbedingungen haben hierbei nur einen geringen Einfluss auf die Materialien selbst. Die heutige Universitätslandschaft bietet den Studierenden eine Vielzahl an Möglichkeiten zu sozialen Kontakten und gruppendynamischen Prozessen. Diese Prozesse und Möglichkeiten zur Interaktion und Zusammenarbeit müssen bei der Einführung eines E-Learning-Systems, d.h. der Plattform bzw. der Applikationen für die Lernenden, Lehrenden und Betreuer beachtet werden. Hier werden im Besonderen interaktive[3] sowie nachlaufende[4] Kommunikationsmöglichkeiten gefordert.

Im Gegensatz dazu, beeinflussen die angesprochenen pädagogischen Randbedingungen die zu erstellenden und zu verwendenden Lehrmaterialien wesentlich. Geht man

[1] Animierte Folie, Demonstratoren und multimediale Anteile wie Filme und Screenvideos von komplexen Sachverhalten stellen hier den Hauptteil der Last dar

[2] Metadaten, welche die Eigenschaften und Einsatzszenarien des Materials beschreiben

[3] zum Beispiel Videoconferencing, Telefon oder Chat

[4] zu Beispiel E-Mail oder Forensysteme

von der vorlesungsbasierten Universitätslehre mit Übungen und Seminaren aus, so ist der pädagogische Ansatz, der hier verfolgt wird, stark durch die direkte Wissensübermittlung durch die Dozenten geprägt. Das Einführen eines E-Learning-Systems stellt dagegen das pädagogische Konzept des selbstgesteuerten Lernens in den Vordergrund. Es ist einsichtig, dass dieses, dem ursprünglichen Ansatz nahezu diametral gegenüberstehende Konzept, eine wesentliche Überarbeitung existierender beziehungsweise das Erarbeiten neuer Lehrmaterialien erfordert.

Eine notwendige Folgerung aus dem dargelegten Sachverhalt ist, dass der durch diese Randbedingungen entstehende Aufwand von den einzelnen Institutionen nicht oder nur unzureichend zu leisten ist. Eine Lösung für dieses Problem ist der Austausch und die Wiederverwendung von bereits existierenden Materialien. Dieses ist besonders dann eine Erleichterung, wenn man nicht nur eigene Materialien, sondern auch Materialien anderer Institutionen verwenden kann. Um dieses Vorgehen zu ermöglichen, müssen die Materialien jedoch hohen Anforderungen[5] hinsichtlich Qualität und Format genügen.

Um die Autoren von Lehrmaterialien bei deren Erstellung, unter Einhaltung der genannten Randbedingungen, zu unterstützen, stellt die vorliegende Arbeit ein Autorensystem vor, das, basierend auf „Office“-Standardprodukten, die Erstellung dieser Materialien für den Autor wesentlich erleichtert. Es erlaubt dem Autor, die Materialien in einem herkömmlichen Text-System zu erstellen und in modularisierter und annotierter Form auszugeben. Das System unterstützt den Autor damit nicht nur bei der Modularisierung, sondern auch bei der Erstellung der notwendigen Metadaten.

Sind alle Randbedingungen für die Materialien erfüllt, muss eine Plattform geschaffen werden, die die Suche und den Zugriff auf diese Materialien für Lehrende, Autoren und Lernende anbietet. Zusätzlich zum Zugriff auf die Materialien soll diese Plattform auch die notwendigen nutzerseitigen Applikationen, wie Lern- und Bildungsportale sowie Autorensysteme, und ihre Anforderungen, z.B. Personalisierung, Nutzermobilität und Abrechenbarkeit von Inhalten, unterstützen. Es soll gewährleistet werden, dass eine möglichst breite Palette von E-Learning Anwendungen an das Serversystem angebunden

[5] Die Anforderungen die hier erfüllt werden müssen, sind ein modularer Aufbau sowie eine strukturierte und weitgehend eindeutige Beschreibung dieser Module

oder in das Serversystem integriert werden kann. Dieses ist zum Einen förderlich für die nachhaltige verteilte Nutzung der erstellten „Inhalte“, zum Anderen eröffnet sich ein Einsatzspektrum weit darüber hinaus. Als ein Fallbeispiel ließe sich der verteilte Zugriff auf ein virtuelles Labor nennen. Eine ebenso denkbare Möglichkeit ergibt sich für Studenten aus der, auch für ihre Materialien, Kommunikationskanäle und Werkzeuge, vereinfachten Bildung von ad-hoc Lerngruppen, welche sich interessen- und inhaltsgetrieben, durch das System unterstützt, spontan zusammenfinden.

Ein Vorschlag für eine solche Plattform, welche zusätzlich noch die Möglichkeit der dezentralen Administration und Installation bietet, wird in der vorliegenden Arbeit vorgestellt.

Das vorgeschlagene System ist hierbei so beschaffen, dass es weitest möglich Betriebssystem- und Netztechnologie beziehungsweise von der Netzanbindung unabhängig dezentral installiert und genutzt werden kann. Die einzelnen Systeme bilden ein Netz dezentraler Server und ermöglichen einen verteilten Zugriff auf die Ressourcen der Einzelsysteme. Ziel ist es, die Interaktion sowie den Datenaustausch der Server unter der Berücksichtigung mobiler Nutzer und deren Applikationen netztransparent zu ermöglichen. Um diese Netztransparenz zu erreichen, ohne dass eine Vielzahl von Voraussetzungen im Netz der jeweiligen Institution erfüllt sein müssen, basiert die Kommunikation zwischen den Anwendungsinstanzen des vorgeschlagenen Systems auf einem Overlay-Netz.

Dieses Overlay-Netz wird in der vorliegenden Arbeit durch ein Peer-to-Peer System[6] gebildet. Der Einsatz eines solchen Peer-to-Peer basieren Overlay-Netzes als Basiskommunikationsmechanismus wirft jedoch die Frage nach der Skalierbarkeit des Gesamtsystems auf. In der vorliegenden Arbeit wird dargestellt, wie das System unter den beschriebenen Randbedingungen im Bezug auf die Skalierbarkeit des Overlay-Netzes optimiert werden kann. (Zu der Problematik der Skalierbarkeit von Peer-to-Peer Netzen siehe auch [25]).

[6] Als System wird hier JXTA, ein Basis-Peer-to-Peer System genutzt

1.2 Motivation

Die Verwaltung, Verteilung und Nutzung von Lehrmaterialien in elektronischer Form stellt die Anwender bereits heute vor eine große Herausforderung. Der ständig schnellere Wandel der Inhalte, deren stetig sich steigernde Spezialisierung und die wachsenden Qualitätsanforderungen führen zu einer Modularisierungs- und Annotationsnotwendigkeit der Materialien. Die sich aus diesem Konzept ergebende Möglichkeit, die erstellten Materialien in anderen Kontexten wiederzuverwenden, bringt viel Vorteile aber auch Probleme, sowohl technischer als auch konzeptioneller Art, mit sich. Zum Einen muss es ermöglicht werden, diese modularisierten Inhalte bereitzustellen und zu finden, zum Anderen stellt die Anforderung der Wiederverwendbarkeit noch einmal höhere Ansprüche an die Materialien, besonders an deren zusätzliche Bechreibung mittels Metadaten.

Die Abbildung der inhaltlichen Zusammenhänge und die Suche nach Modulen wird hierbei heute auf sehr unterschiedliche Art und Weise realisiert. Diese Realisierungen sind dabei jedoch auf die jeweilige Plattform und deren Materialformate festgelegt. Es existieren zwar sogenannte Import- und Export-Filter, deren Verwendung jedoch nicht immer zu den von den Nutzern erwarteten Ergebnissen führen.

Eine weitere Anforderung, welche sich aus dem Wandel der Universitätslandschaft als dringende Notwendigkeit ergibt, ist die Möglichkeit der Vermarktung der Materialien. Auf diesen Teilbereich wird jedoch selten in integrierter Form Wert gelegt. Abrechnungsfunktionen und Zugriffskontrollen müssen dann außerhalb der Plattform abgehandelt und administriert werden. Dies ist gerade im Hinblick auf ein deutschland- oder sogar europaweit genutztes Netz an Anbietern kaum realistisch.

Betrachtet man darüber hinaus die zunehmende Mobilität [7] der Lernenden und deren sich immer schneller entwickelnden Anspruch auf ein „Learn Anywhere, Anytime“, so kann auch dieser nicht allgemein durch die existenten zentralisierten Lösungen befriedigt werden.

[7] Auf den Begriff der Mobilität und dessen Bedeutung im Rahmen des Kontextes dieser Arbeit wird in Kapitel 3.1 eingegangen

Aus diesem Problemfeld heraus erwächst die Notwendigkeit einer Architektur, die die Beteiligten in die Lage versetzt, diese Nachteile ohne einen großen Zusatzaufwand zu überwinden.

1.3 Zielsetzung und Abgrenzung

Das vorgeschlagene System soll die dezentrale Organisation und den Zugriff auf bereitgestellte Daten, Metadaten und Dienste, die zur Unterstützung von E-Learning Anwendungen nötig sind, stukturiert unterstützen beziehungsweise ermöglichen. Dieses soll unter der Maßgabe der Zugriffskontrolle und Abrechenbarkeit der Angebote realisiert werden. Zudem liegt ein weiterer Hauptaspekt auf der problemlosen, das heißt ohne tieferes Administrations- oder Systemwissen, Teilnahme an dem vorgeschlagenen Verbund.

Das vorgeschlagene System geht hierbei nur am Rand auf das eigentliche Content-Management oder die zur Darstellung benötigten Anwendungen ein. Das System bietet vielmehr die Management-, Transport- und Kommunikationsinfrastruktur, welche die Grundlage für Portal- und Content-Management-Lösungen sind. Der Vorteil liegt in der Abstraktion von spezifischen unterliegenden Content-Management- oder Datenbanksystemen. Ein zentraler Aspekt der Arbeit ist es, aufzuzeigen, wie diese Systeme und Anwendungen auf einfache Weise an das vorgeschlagene System angebunden werden können und somit eine Zusammenarbeit der Systeme zu ermöglicht werden kann. Im Rahmen der prototypischen Implementierung werden Ansätze für die Anbindungen von Präsentationsapplikationen und Content-Management-Systemen (CMS) näher betrachtet.

Bezug nehmend auf die Aufwands-Problematik der Inhaltserstellung wird ferner eine Variante der Lehrmodulerstellung vorgeschlagen, welche sich in das hier präsentierte System eingliedert. Das aufgezeigte Werkzeug soll keine vollständig allgemeingültige Lösung darstellen. Es dient im Rahmen der vorliegenden Arbeit einem Proof-of-Concept für ein solches Vorgehensmodell. Die Applikation zur Erstellung und Modularisierung

von Lehrmaterialien trägt maßgeblich dazu bei, die am Ende gegebenen Abschätzungsansätze zum Kommunikationsaufwand des Systems und die notwendigen Basisdaten über Modulgrößen und deren Metadatengrößen angeben zu können.

Die Arbeit geht hierbei jedoch nicht tiefer auf die Metadaten-Struktur selber oder spezielle die Suche unterstützenden Strukturierungsmöglichkeiten für die erstellten Lernmodule ein. Es wird angenommen das solche Strukturierungsmöglichkeiten existieren und die Suche nach passenden Lehrmaterialien wesentlich unterstützen können.

1.4 Gegenstand der Dissertation

Zur Lösung der geschilderten Probleme wird in der vorliegenden Arbeit eine neue der Gesamtproblematik rechnungtragende Architektur für ein dezentrales Server-System zum Austausch und der Bereitstellung solcher Materialien und anderer E-Learning Anwendungen unterstützendender Informationen entworfen. Die Plattform ist hierbei so beschaffen, dass sie zugleich auch weitergehenden Anforderungen aus dem Bereich des E-Learning unterstützend gerecht wird. Hierzu zählen vor allem auch die Unterstützung von Nutzerprofilen und der Personalisierung. Hierbei wird im besonderen auch aufgezeigt, wie dieses mit unterschiedlichsten Anwendungen unter der Randbedingung des Schutzes der personenbezogenen Daten der Nutzer möglich ist.

1.5 Struktur der Arbeit

Die folgenden Kapitel der Arbeit sind wie folgt strukturiert: Kapitel 2 gibt einen Überblick über die technischen Grundlagen der Netze, der verteilten Systeme und des E-Learnings, soweit diese für die vorliegende Arbeit relevant sind. Es gibt ferner Einordnungen und Bewertungen des Standes der Technik insoweit diese Punkte Einfluss auf die nachfolgenden Kapitel haben. Ausgehend davon wird in Kapitel 3 erarbeitet, welche Randbedingungen und Einsatzszenarien sich aus Literatur und Erfahrungen ergeben

und in die weiteren Betrachtungen und den Entwurf des Server-Systems als Basis einbezogen werden müssen. Auf die daraus hervorgehende Teilproblematik der Inhaltserstellung wird in Kapitel 4 eingegangen. Es wird eine Werkzeugkette zur standardkonformen Lehrmodulerstellung entwickelt und vorgestellt. Die Hauptproblematik, der Architektur des verteilten Server-Systems und ein Bewertungsansatz, wird in den restlichen Kapiteln behandelt. Hierbei werden in Kapitel 5 der Architekturentwurf und in Kapitel 6 die zum Funktionalitätsnachweis nötige prototypische Implementierung beschrieben, sowie, darauf aufbauend, Ansätze zu einer technik- und funktionalitätsbezogenen Bewertung der Architektur. Eine Zusammenfassung der Arbeit und ein Ausblick auf weiterführende Fragestellungen werden in Kapitel 7 gegeben.

Die Beiträge in der vorliegenden Dissertationsschrift wurden im Rahmen der Forschungsprojekte ITO (BMBF-Projekt) CANDLE (EU-Projekt des 5ten-Rahmenprogrammes) und DAIDALOS (EU-Projekt des 5ten-Rahmenprogrammes) iniziiert.

Kapitel 2

Grundlagen

In den folgenden Abschnitten werden die zum Verständnis der Arbeit notwendigen Grundlagen wiedergegeben. Hierbei wird auf die Teilbereiche „Technische Grundlagen des E-Learnings“, „Peer-to-Peer-Systeme“ sowie heutige Netzprotokolle und Kommunikationsmechanismen in verteilten Systemen eingegangen. Auf Grund der Komplexität der einzelnen Gebiete wird jeweils nur ein Überblick mit entsprechenden Verweisen auf weiterführende Literatur gegeben. In den Fällen, in denen die Literatur keine ausreichende Klarheit erreicht, werden Konzepte und Definitionen in größerer Tiefe gegeben.

2.1 Technische Grundlagen des E-Learning

> *Die Halbwertzeiten von Wissen verkürzen sich immer weiter. Das Erlernte muss laufend ergänzt werden. Dabei sind Lehr- und Lerngewohnheiten einem ständigen Wandel unterworfen. Das Internet bietet das Potenzial, die weltweite Wissenschaftsgemeinde noch stärker zusammenzuführen.*
> *Wirtschaftsunternehmen begleitet es auf dem Weg zur lernenden Institution, die mit ihrem Wissen das Kapital bildet für den erfolgreichen Marktauftritt. Ihre Grundlagen findet die wachsende Wissenskultur in der IT-Technik. Virtuelle Universitäten und Klassenzimmer entstehen, E-Learning-Plattformen öffnen den Zugang zu explorativem Lernen und kooperativem Arbeiten. Künftig sollen Lernsysteme zur Verfügung stehen, die außer dem Vorwissen und dem Erfahrungshorizont des Lernenden auch seinen individuellen Lernstil berücksichtigen.*
> *Aus: [Flyer zu "GI Fachtagung zum Thema E-Learning (2002)"]*

Der Einzug von E-Learning-Technologien in die Aus- und Weiterbildung begann schon in den 90er Jahren mit Business-TV und Computer Based Trainings (CBT). Mit der Etablierung des Internet kamen später die Web Based Trainings (WBT) hinzu.

Diese Technologien wurden in ihrer Anfangszeit hauptsächlich von Firmen zur dezentralen Mitarbeiterschulung eingesetzt. Sehr früh fanden sich dann, mit der Einführung von Applets[1][63] sowie der Macromedia Flash Technologie, immer mehr interaktive Webseiten mit denen Forschungsergebnisse und komplexe Sachverhalte für die Lernenden „erfahrbar“ dargestellt wurden. Zunehmend flossen diese Inhalte auch unterstützend in die Lehre an den Universitäten ein. Mit der Verbreitung des Einsatzes dieser Technologien wurde auch der hohe Aufwand offenbar, mit welchem deren Erstellung verbunden ist.

Betrachtet man die Entwicklung des E-Learning-Anteils an der Lehre und die Aktivitäten der Forschung in diesem Bereich wie in den Bildern 2.1 und 2.2 dargestellt, so werden wichtige Randbedingungen für einen nachhaltigen Einsatz dieser Lehr- und Lernform offenbar.

Hierbei ist anzumerken, dass der rechte Teil der Entwicklungskurve aus Abb. 2.1 wesentlich von Außen beeinflusst werden muss. Eine selbstständige Entwicklung in eine

[1] Applets sind clientseitig ausgeführte Programme innerhalb von Web-Seiten in der Sprache JAVA

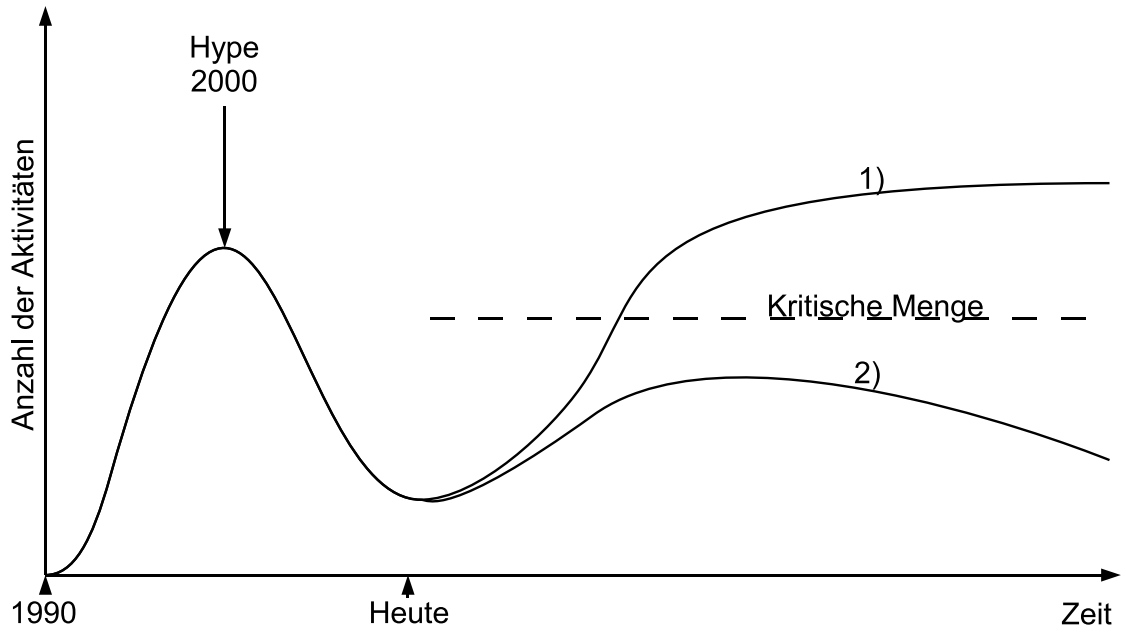


Abbildung 2.1: Vom E-Learning-Hype zur Nachhaltigkeit

Steigerung der Aktivitäten im diesem Bereich oder des verstärkten Einsatzes von E-Learning ist nicht absehbar. Die äußeren Reize können hierbei durch die Institution gegeben werden, wie etwa monetäre oder öffentlichkeitswirksame Anreize der Institution gegenüber den Anwendern (Lehrenden wie Lernenden). Es gibt jedoch auch Faktoren, welche durch die Hersteller der Software bzw. durch die Funktionalität der eingesetzten Werkzeuge beeinflusst werden. So kann die Akzeptanz der Plattform oder Anwendung in den Institutionen mit effizienten Applikationen zur Unterstützung der Lehrenden bei der Erstellung von E-Learning-Materialien gesteigert werden. Ebenso großen Einfluss wie die Autorensoftware zur Erstellung der Inhalte hat die Verständlichkeit und Durchgängigkeit der Werkzeugkette, der verwendeten, beziehungsweise zu verwendenden E-Learning-Umgebung insgesamt.

Denkt man diese Ausführungen konsequent weiter, so ist es ohne Zweifel von besonderer Wichtigkeit, Anreize für alle Teilnehmer einer solchen Lernumgebung zu schaffen. Gelingt es nicht, Lernende, Institutionen und gerade auch die Lehrenden von den Vorteilen und dem möglichen Einsparpotenzial eines nachhaltigen E-Learning-Anteils in der Lehre zu überzeugen, wird es keine Nachhaltigkeit in der Nutzung geben. Eine nachhal-

tige Akzeptanz des E-Learning ist nur dann zu erreichen, wenn sich eine kritische Masse an Anwendern und Material bilden kann (Kurve 1 in Abb. 2.1). Als Auswirkung einer zu flach verlaufenden Steigerungskurve (Kurve 2 in Abb. 2.1) des E-Learning-Aufkommens ergibt sich folglich die Gefahr des „Einschlafens“ des E-Learning-Einsatzes.

Zusammenfassend ist zum Erreichen eines tragfähigen, breiten Einsatzes von E-Learning nötig, die Einhaltung der folgenden Aspekte sicher zu stellen:

Die Schwellen, die heute ein Einsteigen in LMS[2] -Lösungen, multimediale Anteile und so weiter behindern, müssen so niedrig wie möglich ausgebildet werden.

Dies erfordert die Entwicklung weiterer Werkzeuge gerade in den Gebieten der Inhaltserstellung (Authoring) sowie des Content-Managements. Letzteres besonders im Hinblick auf die Verfügbarkeit existierender Materialien, unter Berücksichtigung von Copyright-Randbedingungen, in einer Breite, die als Anreiz interessant ist.

Das Standardisieren eines einheitlichen Formates für die Inhalte, um die verfügbaren Materialien dann auch nutzen und austauschen zu können, ist dazu eine Voraussetzung. Ferner wird, um bereits existierende Inhalte integrieren zu können, die Möglichkeit von Formatumwandlungen notwendig sein.

Ebenso notwendig zu beachten sind ausreichende Beschreibungen der Materialien mit Metadaten und, soweit überhaupt pädagogisch möglich, die Materialien „multi-purpose“ zu befähigen, das heißt unabhängig von der Zielgruppe zu erstellen. Die Komponenten, welche für ein solches System notwendig sind, beziehungsweise in ihm enthalten sein sollten, finden sich in Abb. 2.3

2.1.1 Modularisierung von Lehr- und Lernmaterialien

Das Modularisieren von Lehrinhalten stellt hohe Anforderungen an die zu erstellenden Module und ihre Metadaten, insbesondere wenn die erstellten Module nicht nur in ihrem originären Kontext, sondern auch in anderen inhaltlichen und pädagogischen Zusammenhängen eingesetzt werden sollen. Elf Anforderungen, welche ein Lernmodul erfüllen

[2] Learning Management System

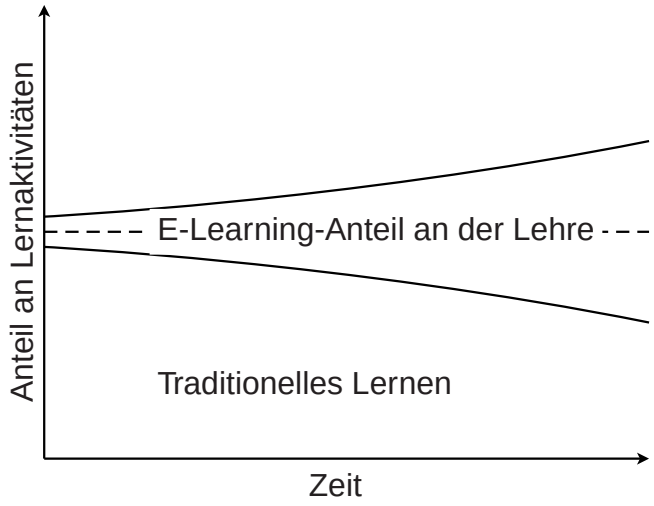


Abbildung 2.2: Entwicklung des E-Learning-Anteils in den Hochschulen

muss, um die geforderten Eigenschaften zu besitzen, wurden in [31] spezifiziert. Werden diese Randbedingungen eingehalten, ist es möglich, einzelne Module auszutauschen und wieder zu verwenden. In Kürze lassen sich diese elf Eigenschaften angeben als:

Formalisierung: Das Beschreibungssystem für die Inhalte muss so gestaltet sein, dass die maschinelle beziehungsweise die automatische Verarbeitung möglich ist. Dies bezieht sowohl Struktur- als auch Metadateninformationen mit ein.

Pädagogische Flexibilität: Die Anwendbarkeit von Lernmaterialien unter verschiedenen pädagogischen Randbedingungen. Das heißt beispielsweise sowohl in Skript, in der Vorlesung als auch in der Nachbereitung anwendbar (Multi-Purposing).

Explizite Typisierung der Module: Die Aufgabe sowie die Struktur und Einbettung des Moduls innerhalb seines Lernkontextes muss eindeutig beschreibbar sein.

Vollständigkeit der Beschreibung: Das Beschreibungssystem muss in der Lage sein, alle das Modul betreffenden Informationen zu enthalten. Das heißt, deren Struktur, die Inhalte, das pädagogische System sowie die zugeordneten Aktivitäten und dessen Position innerhalb des Arbeitsablaufes der Lerner sollen gespeichert werden können.

Reproduzierbarkeit der Ergebnisse: Eine genaue Einordnung des Moduls, das heißt zum Beispiel dessen Lernziele, muss möglich sein, um eine Reproduzierbarkeit der Ergebnisse zu erreichen.

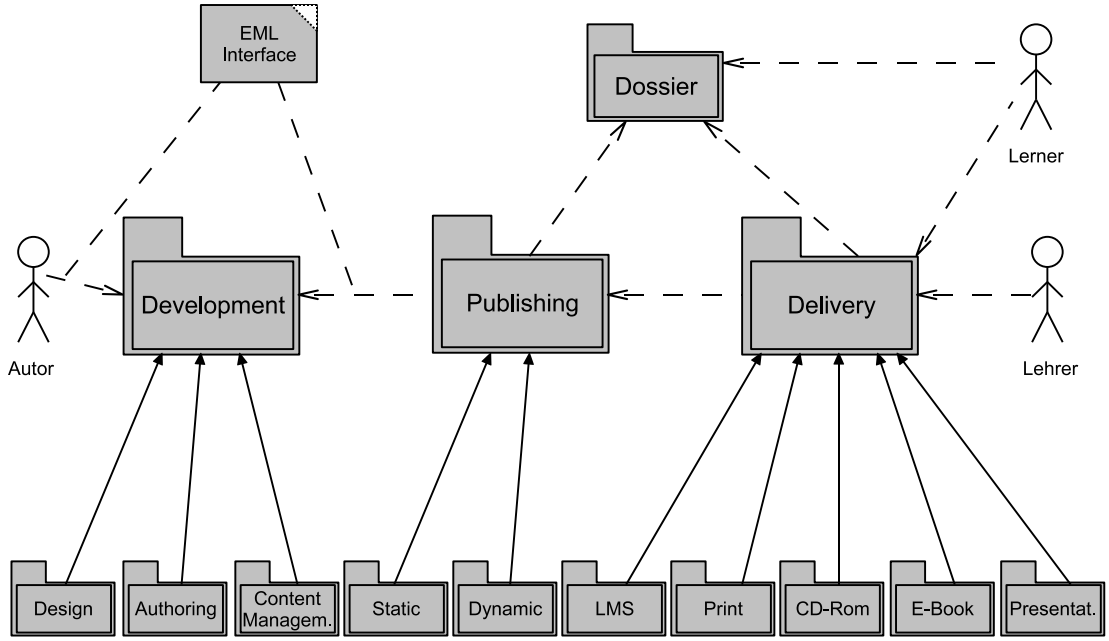


Abbildung 2.3: Referenzarchitektur eines durchgängigen E-Learning-Systems (aus [31])

Personalisierbarkeit des Moduls: Die dem Modul zugeordneten Metadaten sollten es der darstellenden beziehungsweise der auswählenden Software erlauben, die Materialien anhand der Profildaten des Nutzers an dessen spezielle Anforderungen anzupassen. (zum Beispiel: Vorwissen, Umfeld, Lernmethodik des Lernenden)

Medienneutrale Inhaltsablage: Die Inhalte, welche das Modul enthält, sollten in einer Form gespeichert sein, die es erlaubt, sie auf beliebigen Ausgabegeräten mit gleich bleibender Qualität darzustellen.

Interoperabilität und Belastbarkeit (Haltbarkeit): Die Modulbeschreibung muss einer Qualität entsprechen, die die Nachhaltigkeit der Nutzung in kommenden Lehr- und Lernlandschaften sicherstellt. Hierzu gehört im Besonderen die Trennung zwischen der gewählten Darstellung und deren Verarbeitung. Wird dieser Ansatz konsequent verfolgt, so kann davon ausgegangen werden, dass es auch zukünftig möglich sein wird, die Beschreibungen zu konvertieren und zu verwenden.

Kompatibilität zu existierenden Standards: Existierende Standards der Beschreibung von modularen Inhalten und Strukturen sowie der pädagogischen Zuordnungen müssen berücksichtigt werden, um eine Abbildung in diesen zu ermöglichen.

Wiederverwendung: Ausreichende Beschreibung der Module, die deren Wiederverwendung ermöglichen. Hierbei ist besonders darauf zu achten, dass es möglich ist, Module zu identifizieren, zu isolieren und wenn nötig zu dekontextualisieren, um sie dann in anderen Kontexten wieder einsetzen zu können.

Unterstützung aller Phasen der Modul-Lebenszyklen: Für alle Lebensphasen eines Moduls (Erstellung, Änderung, Anpassung, Archivierung,. . .), muss das Beschreibungssystem die notwendige Unterstützung und Ausdrucksmöglichkeiten bereitstellen.

Ebenso werden drei Merkmale für die Klassifikation von Lernmodulen angegeben, welche in den Metadaten mindestens abgebildet werden sollen. Diese sind:

- Die Klassifizierung oder Typisierung der Module in einem semantischen Netz, das von einem pädagogischen Metamodell oder einem entsprechenden ontologischen System abgeleitet wurde,
  dazu
- eine Beschreibung des „Einsatz“-Rahmens, der die Zusammenhänge zwischen den einzelnen Modulen angibt, sowie
- die Beschreibung der Struktur der Inhalte und der Art der verschiedenen Module, die geeignet ist, den Einsatzbereich derselben eindeutig festzuhalten.

Die Projekte CANDLE[3], ITO[4] und Metacoon[5] sowie MMDB[6] sehen eine in der Basis gleiche Strukturierung der Lehrmodule vor. Die Kompositionen dieser Module unterscheiden sich nur in der Komplexität der zu erstellenden zusammengesetzten Inhalte, vergleiche Abb. 2.4.

Die Betrachtung zeigt, dass verbreitet eine solche Struktur für Lehr- und Lernmaterialien angenommen wird. Einzig die Problematik der Granularität, d.h. die Menge an pro Modul dargebotenem Inhalt, bleibt hier offen. Der verbreitetste und viel versprechendste Ansatz, die Granulariät der einzelnen Module zu „definieren“, bestimmt diese nicht

[3] IST-EU Projekt ”Collaborative And Network Distributed Learning Environment”
[4] BMBF Projekt ”Information Technology Online”
[5] Lernportal
[6] Multimedia Datenbank - Essen

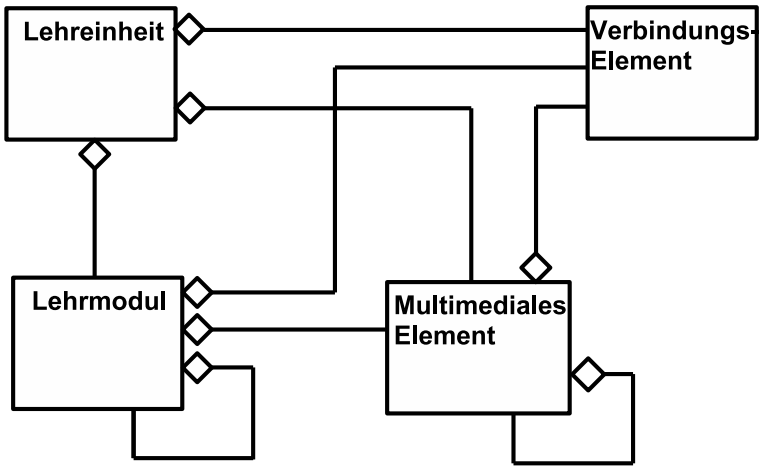


Abbildung 2.4: Basis-Struktur der Lehrmodule

über die Speichergröße, sondern über die Abgeschlossenheit und Menge der erklärten Begriffe und Probleme. Ein solches „Wissensdomänen“-Konzept wurde z. B. im Projekt CANDLE [46] erarbeitet und bewertet.

Shareable Content Object Reference Model (SCORM)

Der heute wohl wesentlichste Standard für Lehrplattformen und deren „interworking“ ist SCORM. Der Name steht für „Shareable Content Object Reference Model“ und wurde vom DOD[7] erarbeitet. Der SCORM-Standard beschreibt hierbei alle notwendigen Eigenschaften der Module, ihre Beziehung untereinander und die Beziehung zum Server. Das Hauptaugenmerk liegt hier auf Eigenschaften, die für einen Austausch der Module wesentlich sind, sowie den notwendigen Schnittstellen für deren Nutzung.

Die in der vorliegenden Arbeit entwickelte Plattform bietet die Möglichkeit, mit SCORM-konformen Materialien zu arbeiten und mit SCORM-konformen Systemen zusammenzuarbeiten. Sie unterstützt durch ihre Konstruktion und Eigenschaften diese Art von Materialien und Systemen sogar in besonders hohem Maß, da sich die in SCORM definierten Schnittstellen zum Zugriff auf ein LMS und dessen Inhalte sehr einfach auf die Strukturen des Systems abbilden lassen.

Eine tiefere Betrachtung des allgemeinen Begriffes „Modul“ sowie des speziellen Begrif-

[7] DOD - Department of Defence

fes des „Lehrmoduls“ wird im Folgenden gegeben. Diejenigen Aspekte der Lehrmodule, welche im Rahmen der vorgestellten Architektur wichtig sind, werden eingehender diskutiert.

Definition des Modul-Begriffes sowie Abgrenzung und Einordnung

Da der Begriff „Modul“ im wesentlichen durch die Fachgebiete der Soft- und Hardware-Entwicklung vorgeprägt ist, soll im Folgenden, bevor auf die hier genutzte technische und methodische Modul-Definition eingegangen wird, eine Auflistung der Gemeinsamkeiten und Unterschiede der jeweiligen Modul-Definitionen gegeben werden.

Module werden in allen Bereichen als in sich abgeschlossene funktionale Einheiten eines größeren Systems verstanden [42]. Hierbei ist ein Software-Modul ein Teil eines Programms und führt eine bestimmte Aufgabe aus. In ähnlicher Weise ist ein Hardware-Modul eine funktional abgeschlossene Einheit, welche an ein System angeschlossen, angebunden beziehungsweise in das System integriert werden kann. Um diese Anbindung eines Moduls an ein Hard- oder Softwaresystem zu ermöglichen, müssen Vorbedingungen eingehalten werden. Diese werden im Allgemeinen als Schnittstellen bezeichnet. Anhand der Beschreibung der Schnittstellen kann dann die Möglichkeit der Anbindung festgestellt werden. Passen die Schnittstellen in allen relevanten Parametern zusammen, wird das Modul innerhalb des Programms oder Systems dann in der vorgesehenen Weise funktionieren. Dass sich die Schnittstellenbeschreibungen beziehungsweise Schnittstellendefinitionen von Soft- und Hardwaresystemen dabei grundlegend unterscheiden, ist evident und soll hier nicht weiter vertieft werden.

Bei den Lehrmodulen gilt im Grundsatz das gleiche Verständnis. Der Raum der Parameter, welcher für einen sinnvollen Einsatz eines Lehrmoduls in einem neuen Kontext beschrieben werden muss, ist jedoch im allgemeinen wesentlich größer als bei Hard- oder Softwaremodulen. Wesentlich ist, dass zu den rein technischen Parametern, welche als „harte“ Parameter angesehen werden können, noch eine große Menge an „weichen“ Parametern, wie zum Beispiel pädagogischen oder kontextualen Randbedingungen, beschrieben werden müssen. Die Schnittstellenbeschreibungen sind, im Kontext

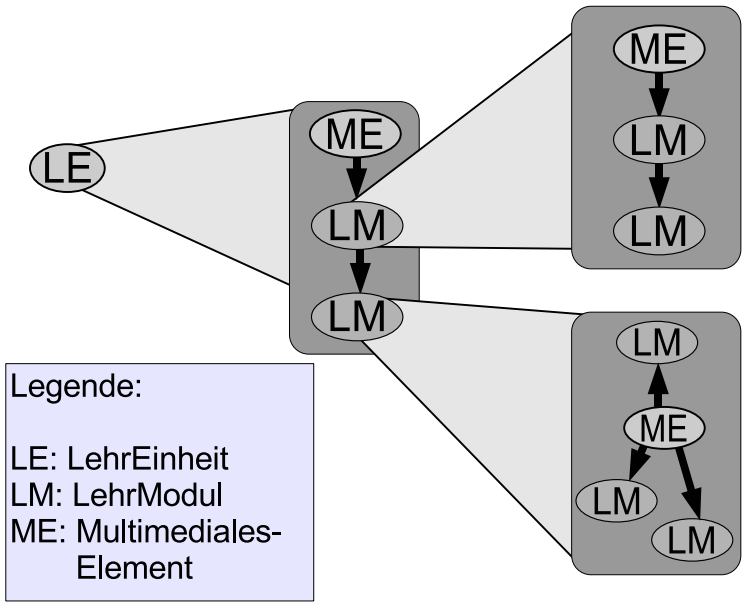


Abbildung 2.5: Zusammensetzung von Lehrmodulen

der Lehrmodule, die Metadaten.

Der Aufbau eines Systems aus Lehrmodulen kann dann wie folgt verstanden werden: Die rein technische Modul-Struktur wird durch eine Menge von Elementen mit festgelegten Beziehungen zueinander abgebildet. Ein Element ist hierbei entweder eine Struktur oder ein Dokument. Dokumente und Strukturen unterscheiden sich jedoch rein äußerlich nicht voneinander, sondern sind nur durch ihren jeweiligen „inneren“ Aufbau zu unterscheiden. Strukturen dienen als Container für Strukturen und Dokumente mit festgelegten Beziehungen, wohingegen Dokumente eine, im Sinne der rein technischen Modularisierung, nicht zusammengesetzte Einheit bilden. Beiden Element-Typen zugeordnet sind gleichermaßen ein Satz von Metainformationen oder Metadaten sowie eine optionale Element-Menge, welche weitergehende Informationen und Materialien für die Nutzer enthalten. Diese Aufspaltung ist in Abb. 2.5 dargestellt.

Die Speicherung dieser Inhalte oder Module kann dann in Content-Management-Systemen stattfinden, welche im anschließenden Abschnitt kurz beschrieben werden.

2.1.2 Content Management

Die wichtigsten Eigenschaften von Content Management Systemen werden nachfolgend aufgezählt und diskutiert. In realen Systemen, welche sich im Einsatz befinden, wurden diese auch realisiert und angewendet. Die Gartner-Consulting Studie „The Emergence of Distributed Content Management and Peer-to-Peer Content Networks“ aus dem Jahr 2001 für Web-Content-Managementsysteme [22] enthält hierzu eine Aufstellung der angesprochenen Parameter. Diese, durch den Gilbane-Report vom März 2004 unterstützte und verifizierte Studie, führt zu folgenden Schlüsseltechnologien:

Einfacher Zugriff auf Inhalte: Das System muss dem Nutzer durch geeignete Mechanismen einen einfachen Zugriff, Suche und Verwendung, auf die abgelegten Inhalte ermöglichen.

Kombinierbarkeit von Inhalten: Die Inhalte sollten in einer Form verwaltet werden, dass eine Kombination verschiedener Materialien zu einem „neuen“ Inhalt möglich ist.

Trennung von Inhalt und Darstellung: Um beim Einsatz der im System gespeicherten Materialien möglichst keine Einschränkungen für deren Einsatz durch die Nutzer durch deren äußere Form zu implizieren, ist eine nicht Layout-getriebene Ablage der Daten notwendig.

Flexibler und schneller Zugriff auf Inhalte: Die Unterstützung bei Wiederverwendung und Rekombination der im CMS enthaltenen Inhalte muss durch das System für die Anwender einfach und flexibel realisiert sein.

Unterstützung bei der Inhaltserstellung: Die Erstellung und Pflege der Inhalte muss durch das System unterstützt und gesteuert werden, um die Konsistenz der enthaltenen Daten sicherzustellen und die Autoren bei ihrer Aufgabe maßgeblich zu unterstützen.

Skalierbarkeit der Systeme: Das System muss in allen seinen Eigenschaften so beschaffen sein, das eine beliebige Erweiterung der Daten- oder Nutzermenge möglich ist.

Die Systeme bleiben jedoch hinter den Erwartungen der Nutzer zurück und ermöglichen keine verteilte Datenspeicherung. Sie verharren vielmehr bei dem Muster der zentralisierten Ablage und des zentralisierten Managements. Damit bleibt die Anforderung nach einem flexiblen, aufwandsarm zu administrierenden System und austauschbaren, wieder verwendbaren sowie dezentral abrechenbaren Inhalten, unerfüllt.

Zum einfachen Auffinden und zu einer effizienten Suche der gespeicherten Materialien werden die Inhalte mit zusätzlichen Informationen, den Metadaten, versehen. Eine kurze Zusammenfassung der wesentlichen Standards für Metadaten im Bereich der Lehrmodule wird im folgenden Abschnitt gegeben.

2.1.3 Spezifikation von Metadaten für Lehrmodule

Die Metadaten der Beschreibung für die Module erfolgt in allen Fällen auf der Basis von XML. Die etablierten Standards, wie Dublin-Core(DC) [27], Learning Object Metadata(LOM) [36] oder SCORM [65] können durch geeignete Umformer ineinander überführt werden. Das Projekt EDUTELLA [15] verfolgt beispielsweise diesen Ansatz.

Die Modul-Metadaten enthalten nicht nur Informationen über die Inhalte des beschriebenen Moduls, sondern darüber hinaus Informationen über pädagogische und strukturelle Randbedingungen, für die dieses Modul ursprünglich entworfen wurde. Zusätzlich enthält jeder Metadatensatz eine ontologische und domänenspezifische Einordnung der Inhalte beziehungsweise des enthaltenen Lernstoffes. Somit ist es möglich, das Modul strukturiert in einen inhaltlichen sowie pädagogischen Kontext einzuordnen und, diesem zugeordnet, in dem System abzulegen und auffindbar zu machen.

Die hohe Anzahl an Dimensionen, die sich durch die große Anzahl nicht immer orthogonaler Beschreibungsparameter für die Suche nach geeigneten Inhalten ergibt, macht die effiziente Suche zu einem wichtigen Forschungsbereich.

2.1.4 Suchmechanismen für Lehrinhalte

Durch die Einordnung der Inhalte in einen kontextspezifischen Rahmen, sowohl pädagogisch als auch inhaltlich gesehen, wird eine strukturierte Suche nach Lehrmodulen durch das System ermöglicht. Der Vorteil einer in dieser Form strukturierten Suche liegt zum einen im geringeren Kommunikationsaufwand für das System und zum anderen in der „Treffgenauigkeit“ der Suche für den Anwender. Die vielversprechendsten Ansätze liefern Ontologie-basierte Verfahren zur Kategorisierung der Module [17, 41].

Eine andere, zusätzliche Möglichkeit die Daten zu strukturieren, kann auch die Einordnung von Daten mittels deren Zuordnung zu einer Expertengruppe sein. Hierbei finden sich Institute zusammen, welche in einem Gebiet Expertise haben und Inhalte für diesen Bereich erstellen und zur Verfügung stellen. Unterhalb der Expertengruppenstruktur sollte dann wieder eine weitere Kategorisierung nach ontologischen Gesichtspunkten angeschlossen sein. Damit kann für den Nutzer der Inhalte eine klare Navigation zu Inhalten seines Interesses ermöglicht werden.

Die beschriebenen Verfahren dienen zur Suche nach einzelnen Modulen. Die modulinternen Strukturen und Abhängigkeiten müssen auf eine andere Art und Weise abgebildet werden.

2.2 Beschreibungen von Modulkompositionen

Um die interne Struktur von Modulen zu beschreiben, bedienen sich die verschiedenen existierenden Systeme unterschiedlicher Strukturbeschreibungsarten.

Als ein Standard ist hier beispielsweise das „Resource Descripton Framework“ RDF [33] zu nennen, welches durch das W3C[8] standardisiert wurde, und heute zur „Semantic Web Activity“ des W3C zählt. Alle Strukturierungsbeschreibungen, welche mit Graphen oder Datenbanken abgebildet werden, können auch in RDF bidirektional transformiert werden. Alle verbreiteten Formate stellen also eine Untermenge der mit RDF beschreib-

[8] World Wide Web Consortium

baren Zusammenhänge dar. Deshalb sollen nachfolgend die wesentlichen Strukturierungsaspekte für Lehrinhalte erwähnt werden.

Eine Strukturbeschreibung für Lehrinhalte muss sowohl die Abfolge und Zusammenhänge der einzelnen Komponenten eines Moduls abbilden, als auch die inhaltlichen, semantischen und pädagogischen Randbedingungen darstellen können. Zu diesem Zweck werden verschiedene Arten von Knoten und Kanten benötigt, welche die jeweiligen Eigenschaften der durch sie abgebildeten Komponenten oder des entsprechenden Zusammenhangs beschreiben. Diese jeweilig zusätzlich benötigten Informationen werden als Attribute auf den Kanten und Knoten gespeichert.

Die Vollständigkeit des Resource Description Frameworks und seine sich stetig weiter ausdehnende Verbreitung in verschiedenen mit den unterschiedlichen Themenbereichen der vorliegenden Arbeit sich ähnelnden Projekten, wie zum Beispiel dem „Semantic-Web“, sowie die Notwendigkeit eines Beschreibungs-Rahmenwerkes für die stark strukturierten darin enthaltenen Daten macht eine Beschäftigung mit RDF sinnvoll.

Alle, im Rahmen der Arbeit, aufgezeigten und genutzten Datenstrukturen lassen sich auch in RDF darstellen oder abbilden. Dieser Umstand kann dann als Ausgangspunkt für zum Beispiel die Transformation der Daten in andere Systeme genutzt werden.

2.3 Netz- und systemtechnische Grundlagen

Die vorliegende Arbeit beschäftigt sich mit der Architektur eines verteilten Systems, welches mit Hilfe eines Kommunikationsnetzes Informationen und Daten austauscht. Dieses Kapitel gibt einen Überblick über die netz- und systemtechnischen Grundlagen, auf welchen das System basiert. Hierzu gehören Aspekte des Kommunikationsnetzes, seine für das System wesentlichen Protokolle sowie die sogenannten Peer-to-Peer Netze. Gerade diese stellen für die vorliegende Arbeit die Basismechanismen der entworfenen Kommunikationsstruktur bereit.

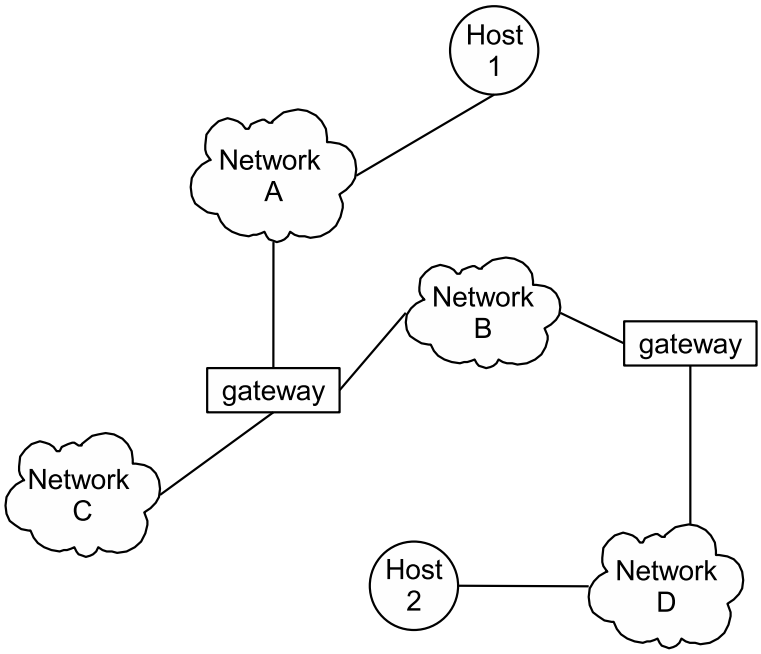


Abbildung 2.6: Ein einfaches Netz

2.3.1 Das Internet und seine wesentlichen Protokolle

Das „Internet“ beziehungsweise die heute allgemein als solches bekannten Protokolle TCP, UDP und IP sind eine Entwicklung der DARPA[9]. Im Folgenden sollen die dort entwickelten universellen Protokolle, auf denen das Internet basiert, kurz beschrieben werden, um das Verständnis der späteren Ableitung von Problemen zu ermöglichen.

Ein einfaches Netz ist in Abb. 2.6 skizziert.

Eine Übersicht findet sich in Tabelle 2.1. Diejenigen Transportprotokolle, welche später in die Architektur des Systems eingehen, sollen dann kurz beleuchtet werden. Die Schichtung dieser Protokolle ist im Abb. 2.7 dargestellt.

In den folgenden Absätzen werden die wichtigsten Protokolle in Kürze beschrieben:

IP - Internet Protocol: Das IP Protokoll stellt den Basiscontainer und dessen Weiterleitungsmechanismen für alle nachfolgend beschriebenen Protokolle dar. Es stellt die Paketstruktur, die Addressierungsmechanismen und die Verkehrslenkungsmechanismen (Routing-Protokolle) für Einzelpakete (Datagram) bereit, in welche die weiteren Proto-

[9]US Defence Advanced Research Proejct Agency

Tabelle 2.1: Basisprotokolle des Internets

Protokoll	Beschreibung
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
IP	Internet Protocol
ICMP	Internet Control Message Protocol
FTP	File Transfer Protocol
TFTP	Trivial File Transfer Protocol
HTTP	Hypertext Transfer Protocol
NFS	Network File System

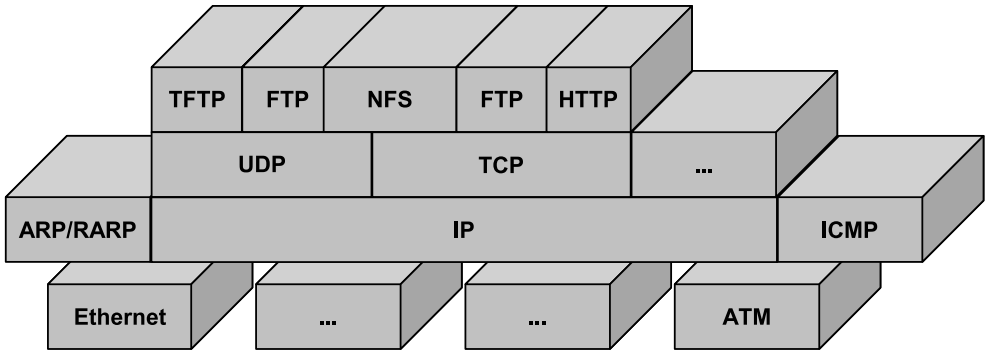


Abbildung 2.7: Die Schichtung der Internetprotokolle

kolle eingebettet sind beziehungsweise welche von ihnen benutzt werden.

UDP - User Datagram Protocol: UPD, das „User Datagram Protocol“ dient zum statusfreien ungesicherten Austausch von Informationen zwischen zwei oder mehr Rechnern beziehungsweise Anwendungen.

TCP - Transmission Control Protocol: TCP, das Transmission Control Protocol ist ein statusbehaftetes gesichertes Datenübertragungsprotokoll auf Strom-Basis. Da es durch seine eingebaute Sicherungsschicht sowie die inhärente Flusskontrolle die Randbedingungen der meisten Anwendungen erfüllt, ist es heute das im Internet meist verbreitete und gebräuchlichste Protokoll.

HTTP - HyperText Transfer Protocol: HTTP, das HyperText Transfer Protocol, ist im Gegensatz zu den beiden vorher erwähnten Protokollen kein Basisprotokoll des Internet, sondern eine Anwendung zum Zugriff auf und Austausch von Daten, welches den

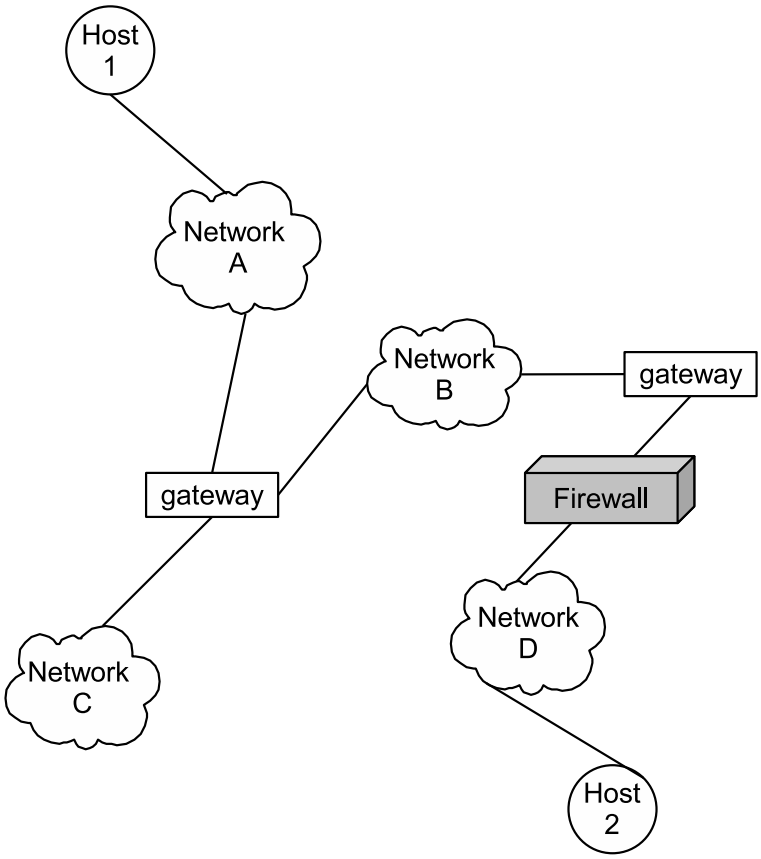


Abbildung 2.8: Einsatz von Firewalls im Internet

gesicherten TCP-Transportmechanismus benutzt.

2.3.2 Sicherheitsmechanismen im Internet

Durch die zunehmende Nutzung des Internets wurden Sicherheitsprobleme offensichtlich und es wurden über die Jahre Mechanismen entwickelt, den Zugriff auf einzelne angeschlossene Rechner oder ganze Netzbereiche einzuschränken. Der Einsatz dieser Mechanismen, wie Firewalls (vergleiche Abb. 2.8), hat jedoch auch den Nachteil, dass Ressourcen nicht mehr frei zur Verfügung stehen oder zur Verfügung gestellt werden können. Hat der Anwender oder Anbieter keinen administrativen Zugang zu den Sicherheitseinrichtungen und kann die benötigten Ressourcen nicht freigeben oder freigeben lassen, so ist es nicht möglich, die Inhalte wie gewünscht zugänglich zu machen.

2.3.3 Verteilte Systeme und deren Kommunikationsmechanismen

Als verteilte Systeme werden Gruppen von Systemen bezeichnet, welche über einen Kommunikationskanal Nachrichten austauschen und diese verarbeiten. Ob die Gruppe hierbei das Client-Server-, das P2P-, das Grid- oder ein anderes Paradigma verfolgt, muss nicht in Betracht gezogen werden. Ein wesentliches Auszeichnungsmerkmal verteilter Systeme ist, das es keine Komponente im System gibt, bei deren Ausfall die Dienstleitung des Gesamtsystems beeinträchtigt ist („No single point of failure“). Die Zusammenarbeit der einzelnen Systeme, das heißt der Austausch und die Verarbeitung der Nachrichten erfolgt dann im allgemeinen nach festgelegten Protokollen oder Schemata. In den folgenden Abschnitten sollen die gebräuchlichsten dieser Mechanismen kurz vorgestellt werden.

Remote Procedure Call Mechanismen

Remote Procedure Call (RPC) Mechanismen sind die Grundlage für den Aufruf von Funktionen auf einem entfernten Rechner über einen Kommunikationskanal. Die wesentlichen Unterschiede der existierenden Standards liegen in deren Eigenschaften hinsichtlich Programmiersprachen und Plattformabhängigkeit sowie dem verwendeten Kommunikationskanal. Im Folgenden werden die vier meist angewandten RPC[10]-Mechanismen kurz beschrieben, vergleiche hierzu Abblidung 2.9

Sun’s Remote procedure call (RPC): Als Basismechanismus der von SUN Microsystems Inc. entwickelten NFS-Plattform wird der im Jahre 1984 ebenfalls von SUN Microsystems Inc., der RPC-Mechanismus eingeführt. Im „Request for Comments“ RFC 1057 [61] ist die Spezifikation dieses Prozedur-orientierten Protokolls zur Nutzung von verteilten Diensten dargelegt.

Secure remote procedure calls: Dieses Protokoll wurde als Erweiterung von RPC in Rahmen von Kerberos, AFS und DFS eingeführt. Es verwendet den Mechanismus DES

[10] RPC wird hier im folgenden als Oberbegriff für derartige Mechanismen genutzt, auch wenn eine Verwechslungsgefahr mit dem von Sun entwickelten als RPC bezeichneten Mechanismus besteht.

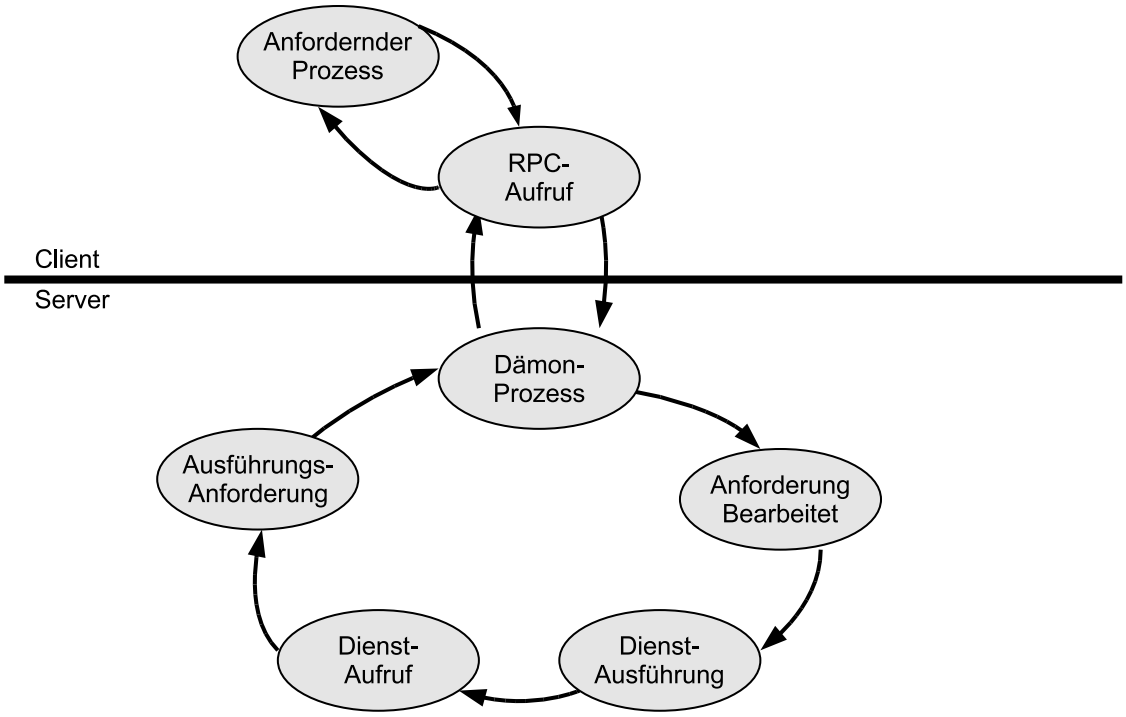


Abbildung 2.9: Grundmechanismus von entfernten Funktionsaufrufen

(Data Encryption Standard) zum Verschlüsseln der RPC-Nachrichten. Die Schlüssel werden hierbei durch einen Key-Server verwaltet. Hierdurch wird eine sichere Übertragung der Daten bei der Nutzung von RPC's gewährleistet und hierdurch der nicht autorisierte Zugriff auf Dienste und Daten durch fremde Rechner eingeschränkt. Die angesprochene Key-Server-Infrastruktur und die dem Kerberos inhärenten Sicherheitsmechanismen können auf der Web-Seite des MIT Kerberos Team [45] und der Kerberos V5 Spezifikation [29] nachgelesen werden.

RMI: RMI (Remote Method Invokation) wurde von Sun als RPC-Mechanismus der Programmiersprache JAVA eingeführt. RMI bietet dieselbe Funktionalität wie RPC, ist in seiner Ausprägung jedoch spezifisch der Programmiersprache Java und ihren Strukturen angepasst.

xmlrpc und SOAP: [71][5] Die Mechanismen „XML-Remote Procedure Call“ (xmlrpc) und „Simple Object Access Protocol“ (SOAP) wurden 1999 vorgeschlagen und 2000 standardisiert. Beide Protokollvarianten gelten heute als Basis der sogenannten Web-Services. Der xmlrpc-Standard stellt hierbei eine Untermenge der Funktionalität des

SOAP-Standards dar. Er ist weniger allgemein gefasst und weniger flexibel als der SOAP-Standard, jedoch ist er für viele Bereiche dadurch leichter zu implementieren und kann viele Anforderungen erfüllen, solange keine komplexen, nutzerdefinierten oder programmiersprachenunabhängigen Datentypen benötigt werden.

Abstraktionsmechanismen zwischen den Protokollebenen:

STREAMS: Streams dienen im Unix System V R4 als Abstraktionsschicht für alle Zugriffe auf beliebige Ressourcen innerhalb eines Systems. Diese Abstraktion wird durch einen Stapel verketteter und kombinierbarer Bearbeitungs- und Filtermechanismen erreicht. Der große Vorteil dieses Mechanismus liegt in dessen Eigenschaft, dass für die Darstellungsebene der Protokolltreiber gegenüber den Treibern der Hardware einheitliche Zugriffsverfahren gegeben sind, welche eine Mehrfachverwendung der verschiedenen Implementierungen ermöglichen. Diese Art der Abstraktion des eigentlichen Transportmechanismus von den darüber liegenden Protokollschichten wird auch innerhalb des hier vorgestellten Systems eingesetzt. Durch die Abstraktion wird erreicht, dass die verschiedensten Dienste wie Simulationswerkzeuge oder virtuelle Laboratorien in die Plattform integriert werden können.

Sichere Transportmechanismen und Protokolle (SSL, STLP und TLS): [16] Die Absicherung von Datenströmen über „unsichere“ Medien und Verbindungen erfolgt heute meist mit standardisierten Protokollen wie dem Secure Session Layer (SSL), dem Secure Transport Layer Protocol (STLP) oder Transport-Layer-Security (TLS)-Mechanismen.

In der vorgeschlagenen Plattform müssen auf Grund der Anforderungen die Eigenschaften dieser Protokolle für die vorgeschlagenen Transportmechanismen nachgebildet werden. Dies ist in der Hauptsache dadurch bedingt, dass die Anforderung nach einer sich unabhängig von der Netzinfrastruktur des Server-Betreibers installierbaren und nutzbaren Plattform in letzter Konsequenz nur die Nutzung der „überall“ verfügbaren Basismechanismen (HTTP, TCP, IP) erlaubt. Die Nutzung von Sicherungsmechanismen würde hier im Regelfall eine problemlose Integration des Server-Systems wesentlich erschweren oder gar unmöglich machen.

Verteilte Filesysteme

Die Informationsverteilung des in der vorliegenden Arbeit beschriebenen Systems ist durch ein verteiltes virtuelles Filesystem realisiert. Die Basis des hier realisierten Filesystems liegt in bereits existierenden verteilten Filesystemen, welche im Folgenden kurz eingeführt werden. Bei der Einführung wird in der historischen Reihenfolge der Entstehung vorgegangen werden. Das letzte vorgestellte Filesystem „Ocean Store“ ist ein in der Forschung und Entwicklung befindliches Filesystem, das der Idee des Filesystems der vorliegenden Arbeit am nächsten kommt.

NFS (Networked File System): Sun Microsystems, Inc. führte 1985 NFS Version 2 [62] im Rahmen des ONC (Open Network Computing) ein, bevor es 1989 Teil des Unix System V Release 4 wurde. NFS basiert auf den Protokollen UDP oder TCP und dem durch BSD4.3 definierten VFS(Virtual File System) Interface. Damit steht die Möglichkeit zur Verfügung, Dateien zwischen Computern mit unterschiedlichen Betriebssystemen auszutauschen. NFS ist heute das gebräuchlichste verteilte Filesystem im Unix-Umfeld. Hierbei wurde es über die Jahre ständig an neu aufkommende Anforderungen angepasst. So wurde im Juli 1992 die Version 3 des NFS-Standards [8], welche gestiegene Performance-Anforderungen erfüllt, und im Dezember 2000 die Version 4 [55, 56], welche Anpassungen für die Weitverkehrsfähigkeit, Sicherheitsmechanismen und eine Überarbeitung des Interfaces enthält, eingeführt.

RFS (Remote File Sharing): [50] Das Remote File Sharing-Protokoll wurde 1987 im Rahmen der Entwicklung von Unix System V Release 3 von AT&T ebenfalls auf Basis des RPC-Mechanismus eingeführt. Ziel der Entwicklung war es, ein verteiltes Filesystem zu schaffen, welches vollständig der Semantik des Unix-Filesystems entspricht. Der Vorteil des RFS ist die Unabhängigkeit vom Transportmechanismus. Dagegen steht allerdings der Nachteil, dass RFS aufgrund seiner Architektur nur zwischen Unix-Systemen eingesetzt werden kann.

AFS (Andrew Filesystem), CODA und DCE/DFS (Distributed Filesystem): [1, 6, 30] Performance- und Sicherheitsaspekte waren die treibenden Anforderungen, die zur Ent-

wicklung der genannten Filesysteme führten. Die Filesysteme AFS, DCE/DFS sowie das Coda-Filesystem sind hierbei historisch miteinander verwandt und wurden mit ähnlichen Zielen und Anforderungen parallel entwickelt. Der wesentliche Unterschied zu den bereits beschriebenen Filesystemen (mit Ausnahme von NFS V4) ist die Unterstützung der folgenden Eigenschaften: AFS, DCE/DFS und Coda stellen ein auch in einem Weitverkehrsnetz performantes, sicheres und ausfallresistentes verteiltes Filesystem dar, welches eigene Sicherheits- und Zugriffsrechtsmechanismen hat und sich auf die Verteilung von Inhalten auf verschiedenen Servern sowie extensives Caching[11] stützt. Das Coda-Filesystem bezieht parallel noch ein persistentes, lokales Caching von Daten auf dem Rechner zur Mobilitätsunterstützung der Anwender mit ein.

SMB (Server Message Block) and CIFS (Common Internet Filesystem): Dieses, von der Firma Microsoft 1995 entwickelte und über NetBios und TCP verwendbare Protokoll erlaubt den Austausch von Dateien über Windows-Rechner hinweg. Im Gegensatz zu den bisher beschriebenen Protokollen beziehen SMB und CIFS nicht nur Verzeichnisstrukturen und Dateien, sondern auch Drucker und ähnliche Ressourcen mit ein.

WEB-DAV (Web-based Distributed Authoring and Versioning): Web-Dav wurde 1999 im RFC 2518 als ein Protokoll zur Pflege der Inhalte auf WEB-Servern entwickelt und standardisiert. Der treibende Faktor für diese Entwicklung war es, das Problem der Wartung und Pflege der Inhalte auf den verschiedensten WEB-Servern über einen einheitlichen Mechanismus zu ermöglichen. Hierbei lagen neben einem an einem „normalen“ Dateisystem orientierten Interface auch die Problematik des verteilten, konkurrierenden Bearbeitens der Inhalte sowie der Zugriffssteuerung auf diese im Fokus der Entwicklungen. Das WEB-DAV-Protokoll erfüllt alle angesprochenen Randbedingungen, indem es, in Erweiterung des HTML-Standards und dem Einsatz von XML-basierten Datenstrukturen, die für ein solches Filesystem notwendigen Funktionalitäten einführt. Mit dieser Art der Realisierung wird jedoch auch ein nicht unbeträchtliches zusätzliches Datenvolumen für die Abwicklung benötigt, die den Einsatz dieses Protokolls innerhalb der DLE nur für die lokale Nutzung sinnvoll erscheinen läßt.

[11] Festplatten-basiertes lokales Zwischenspeichern von Dateien

Das im folgenden vorgestellte Filesystem kann hier als eine auf modernen Kommunikationsstrategien basierte Weiterentwicklung betrachtet werden.

Distributed Storage Systems (Ocean-Store): Ein modernes System zur verteilten Speicherung persistenter Daten auf Peer-to-Peer Basis ist „Ocean-Store“. Das System soll es nomadischen und ubiquitären[12] Anwendungen ermöglichen, sicher und schnell auf die benötigten Daten zuzugreifen. Hierbei werden die Daten für den Nutzer transparent verteilt in den Ocean-Store-Datenpools gespiegelt und verschoben, wenn die Nutzungs-Überwachung dies anhand der Nutzungsmuster für gegeben detektiert. Um die Sicherheit der Daten zu gewährleisten, werden diese verschlüsselt abgelegt. Das System sieht hierbei vor, dass die Daten jederzeit und überall zwischengespeichert werden können. Ein Abrechnungssystem ist dabei jedoch nicht vorgesehen. Eine weitergehende Beschreibung kann in [3] nachgelesen werden.

2.3.4 Authentication, Authorization und Accounting (AAA)

Bereits in den frühen 80er Jahren hat sich die ITU[13] und kurze Zeit später ebenfalls die IETF[14] Gedanken zu einer Standardisierung und Funktionsweise von Authentication, Authorization, Accounting und Abrechnungs-Systemen gemacht. Hierbei sind die Entwicklungen und Anstrengungen die ständig steigende Menge durch AAA-Systeme generierter Daten besser verarbeiten zu können, stark angewachsen. Hierbei wurden, da das ursprüngliche Konzept der AAA-Systeme aus heutiger Sicht wesentliche Aspekte „vernachlässigt“ hat, die Architekturen um die Komponenten Auditing und Charging, das heißt um die Erfassung von Nutzungsdaten und Abrechnungsverfahren sowie deren Bearbeitung erweitert. Diese Architekturen werden heute unter dem Begriff A4C behandelt.

Die folgenden Abschnitte geben eine einführende Erklärung der Begrifflichkeiten.

[12] nomadische oder ubiquitäre Anwendungen sind Anwendungen, welche sowohl als Server oder Client zu jeder Zeit erreichbar oder verwendbar sind.

[13] ITU: International Telecommunication Union

[14] IETF: Internet Engineering Task Force

Authentication

Authentication beschreibt den Vorgang der Feststellung der Identität eines Nutzers. Hierbei können unterschiedliche Verfahren genutzt werden, die je nach Umgebung unterschiedliche Sicherheitsanforderungen der Systembetreiber oder der Nutzer erfüllen. Bekannte Verfahren sind zum Beispiel das Login mit Nutzernamen und Passwort oder Verfahren, die den Einsatz elektronischer Medien (Smart Cards, Mobiltelefon oder Äquivalentem) nutzen, deren Besitz dann die Identifikation erlaubt.

Authorization

Auf Basis der Identität der Nutzer, welche über den Authenikationsvorgang festgestellt wurde, können den Nutzern nun Berechtigungen (Authorizations) zugeordnet werden, auf Grund derer sie Dienste des Systems in Anspruch nehmen oder Dokumente schreiben und lesen können. Diese Berechtigungen können entweder mit der Identität des Nutzers direkt, über dessen Gruppenzugehörigkeit oder spezielle Datensätze, die der Identität zugeordnet sind, geprüft, beziehungsweise zugeordnet werden. Die meisten Systeme wenden hierbei eine Mischform der verschiedenen Verfahren, meist aus historischen Gründen, an. In modernen Kommunikationssystemen, in denen die Sicherheit und der Schutz, zum Beispiel der Persönlichkeit der Nutzer, gewährleistet sein soll, kann jedoch nur die letzte Form der Authorization über Datensätze, sogenannte Zertifikate, angewandt werden. Zum Schutz des Nutzers wird in diesen Fällen ein Pseudonym erstellt, welches als solches keinerlei Informationen zur Person trägt, da es dem ursprünglichen Nutzer nur über zusätzliche Daten seines Heimatsystems zuzuordnen ist. Da somit die Identitäts- oder Gruppen-basierte Berechtigungsprüfung für Fremdsysteme nicht mehr möglich ist, bleibt nur die genannte Option.

Soll nun der Nutzer nicht nur in der Lage sein, für ihn freigegebene Ressourcen der Systeme zu nutzen, sondern daß ihm darüber hinaus die Nutzung dieser Ressourcen auch in Rechnung gestellt werden kann, so müssen Daten erfasst werden, welche eine sofortige oder zeitversetzte Abrechnung erlauben. Der Vorgang der Datenerfassung wird

Accounting genannt und soll im folgenden Abschnitt erläutert werden.

Accounting

Bei Accounting geht es um die Erfassung und Speicherung von abrechnungsrelevanten Nutzungs- und Zugriffsdaten von Nutzern auf angebotene Dienste und Daten. Die Datenerfassung muss hierbei in einer Form stattfinden, die die spätere Zuordnung der Datensätze zu Nutzern und Dienstkosten eindeuting und unabstreitbar macht, und so die Rechnungserstellung ermöglicht. Im Rahmen des in der vorliegenden Arbeit vorgestellten Systems ist diese Zuordnung von Kosten zu Nutzern nur indirekt über die zertifikatsausgebende Instanz, das heißt dessen „Heimatserver“, möglich. Die Abrechnungsdaten werden, da die exakte Identifikation der Nutzer durch die verwendeten anonymen Zertifikate nicht möglich ist, der Heimatinstitution des Nutzers in Rechnung gestellt und durch diese dann an die eigentlichen Kostenverursacher weitergegeben.

Die bisher vorgestellten Mechanismen definieren jedoch noch nicht, wie die erfassten Daten ausgewertet und abgerechnet werden. Dieser Punkt kann in dem zu realisierenden System jedoch nicht unbeachtet bleiben, da eine externe Bearbeitung der Daten und die Erstellung von Rechnungen eine zu hohe Belastung für die Betreiber bedeuten würden. Der nächste Abschnitt soll deshalb kurz auf die Grundlagen der Abrechnung und bekannter Abrechnungssysteme eingehen.

Post-Paid: Bei „Post-Paid“-Verfahren wird erst nach der Nutzung der Dienste, der Inhalte oder der in Anspruch genommenen Leistung bezahlt. Die Abwicklung basiert hier auf vertraglichen Bindungen zwischen den beteiligten Parteien. In diesem Abrechnungsmodus werden die Accountig-Daten gesammelt und die entstandenen Kosten den Nutzern später in Rechnung gestellt. Ein solches Verfahren ist im Rahmen der Arbeit beispielsweise zwischen Institutionen vorstellbar.

Pre-Paid: Bei der Abrechnung im „Pre-Paid“-Verfahren werden die auf Grund der Nutzung entstandenen Kosten sofort von einem vorher bestehenden Konto, auf welchem ein entsprechender Gegenwert vorhanden sein muss, abgebucht. Die sofortige Abrechnung

(Charging) bedingt hierbei, dass das System zusätzliche Leistungen erbringen muss. Da im Regelfall eine „Überziehung“ des Kontos durch den Nutzer nicht in Betracht kommt, muss die Möglichkeit der „Guthabenprüfung“ durch das System bereitgestellt werden. Besonderer Aufwand entsteht durch die notwendige Echtzeitfähigkeit dieser Prozesse. Dieses Verfahren ist zum Beispiel zwischen Institutionen und einzelnen Teilnehmern vorstellbar.

Beide Abrechnungsarten haben Eigenschaften, die sie für den Einsatz im Rahmen des hier vorgestellten Systems, einzeln oder in Kombination sinnvoll machen. Eine grundlegende Beschreibung der Einsatzmöglichkeiten findet sich in [68].

Im Folgenden werden noch kurz die Themen Auditing und Charging umrissen.

Auditing und Charging

Neben der Erfassung müssen die Daten letztendlich dem Nutzer, welcher den abgerechneten Dienst in Anspruch genommen hat, nachweisbar zugeordnet und gegen sein Konto verrechnet werden. Hierbei ist die Problematik der verteilten Daten sowie der pseudonymisierten Nutzer zu betrachten. Die Literatur zeigt eine Vielfalt von Lösungen und Protokollen in diesem Bereich, welche in der jeweiligen Ausprägung im Rahmen des DLE einsetzbar sind. Eine genaue Analyse soll hier nicht vorgelegt werden, es muss jedoch erwähnt werden, dass die entstehende Menge an erfassten Daten mit der Anzahl der teilnehmenden Nutzer stark steigt.

Für das DLE-System wird die folgende Lösung dieser Problematik vorgeschlagen: Die Abrechnungen sollen möglichst nur zwischen den teilnehmenden Institutionen instrumentalisiert und nicht auf alle Nutzer einzeln heruntergebrochen werden. Ordnet man den Abrechnungssätzen noch die Pseudonyminformationen des Nutzers zu, so kann eine nachgeschaltete Abrechnung durch die Institute weiterhin erfolgen. In welcher Form und zu welcher Zeit die Datenübermittlung zwischen den Systemen der Institutionen stattfinden sollten, wird in [68] diskutiert und kann, auf das DLE bezogen, dort nachgelesen werden. Im Rahmen der vorliegenden Arbeit wurden die dort dargestellten Konzepte

um Mechanismen zur Unterstützung der dynamischen Nutzung von, auf nahen Servern, in dortigen Caches, gepufferten Inhalten und deren Abrechnung zu ermöglichen, erweitert.

2.4 Das World Wide Web (WWW) und seine Entwicklungen

Durch die Verbreitung der weltweiten Vernetzung der Wissenschaftsinstitute hat sich nach textbasierten Informationssystemen wie E-Mail, News- und Gopher-Diensten am Anfang der neunziger Jahre das sogenannte World-Wide-Web oder „WWW“ als das Medium zur persistenten und weltweiten Bereitstellung von Informationen herausgebildet und seit den Anfängen massiv weiterentwickelt.

2.4.1 WWW

Der Erfolg des WWW ist dabei hauptsächlich durch seine Flexibilität und den bereitstehenden Möglichkeiten getrieben. Darstellung von Texten, Grafiken und interaktiven Anteilen wie Formularen und eingebetteten Programmen bieten die Möglichkeit, beliebige Inhalte in der jeweilig besten Form zu präsentieren. So wurde das WWW bald zu einer der wichtigsten aber auch zu einer der unstrukturiertesten Informationsquellen der Wissenschaftsgemeinde. Mit der Einführung von weltweiten Suchmaschinen wie AltaVista und Google wurde es zwar einfacher Inhalte zu finden, doch die Möglichkeit einer Strukturierung des WWW wurde bald zum Forschungsgegenstand.

2.4.2 Das Semantic Web

Das „Semantic Web“ wurde 2001 von Tim Berners-Lee vorgeschlagen. Der Hintergrund für diesen Vorschlag war der Wunsch nach einer effizienten Möglichkeit, die Informationen des WWW zu strukturieren und auffindbar zu machen. Darüber hinaus soll es die

Möglichkeit bieten, Dienste, die im Web bereitstehen, zu finden und zu nutzen, sowie den Austausch von Nachrichten von sich im Web befindlichen Agenten ohne menschliche Eingriffe zu ermöglichen.

Um diese Idee Realität werden zu lassen, müssen die Dokumente des WWW mit zusätzlichen Informationen angereichert werden, welche es Maschinen ermöglichen, semantische beziehungsweise ontologische Zusammenhänge zwischen ihnen zu entdecken. Die Beschreibung der Strukturen des „Semantic Web“ wird nach dem RDF- (Resource Description Framework) Standard gespeichert. Zur Beschreibung der ontologischen Zusammenhänge kommen weitere Standards in den Fokus. Kandidaten für die Beschreibung dieser Informationen können in Topic Maps XTM [53] oder Beschreibungssprachen wie DAML-OIL [14] oder OWL [48] ausgeführt sein.

2.4.3 Ontologie-getriebene Suche und Strukturierung

Bei den möglichen Realisierungen von Suchmechanismen im WWW ist die Möglichkeit des Ontologie-basierten Suchens besonders beachtenswert, da hierbei bereits der Kontext, in welchem die angebotenen Inhalte dargeboten werden, in die Strukturierung des Suchprozesses einbezogen werden kann. Zugrunde liegt der Ontologie-basierten Suche die Idee, dass das ganze Suchen in einer derart ausgeprägten Umgebung durch eine Art von dynamischem DNS-Dienst abbildbar ist. Das Auffinden der Informationen kann dann in der gleichen Weise organisiert werden wie die Adressauflösung in heutigen Internet. Eine Anpassung der bekannten Mechanismen muss hierbei allerdings vorgenommen werden. Die herkömmliche DNS-Abfrage zielt nur auf eine Antwort pro Suche ab. Um also die Anforderungen, welche sich aus einer Inhaltssuchanfrage ableiten, abzubilden, ist es notwendig, mehrere, auch zeitversetzte, Antworten auf eine Anfrage zuzulassen. Dieses sollte asynchron geschehen, um die Wartezeit bis zur Übermittlung des ersten Ergebnisses möglichst klein zu halten. Beispiele solcher Mechanismen und Lösungsansätze für die geschilderten Probleme finden sich in [52] und [37].

2.5 Grundlagen von Peer-to-Peer-Systemen

Durch die schnelle Entwicklung der Computer haben sich auch das Profil und das Paradigma ihrer Nutzung über die Jahre verändert. Der ursprüngliche Ansatz des Client-Server Paradigmas, also von leistungsfähigen zentralen Servern, welche Dienste anbieten und schwächeren Klienten, welche die angebotenen Dienste nutzen, nehmen jetzt alle Rechner zunehmend wechselseitig die jeweiligen Aufgaben wahr. Dieses neue Paradigma wird als Peer-to-Peer-Computing bezeichnet. Eine gute Darstellung dieser Entwicklung findet sich in „Von Client/Server zu Peer-to-Peer“ [59].

Da gerade das P2P-Paradigma zukünftig immer wichtiger wird und auch einen zentralen Teil der vorliegenden Arbeit darstellt, sollen diese in den folgenden Abschnitten betrachtet werden.

2.5.1 Paradigmen von Peer-to-Peer-Konzepten

Nach [59] sind die folgend genannten Paradigmen die Leitkonzepte heutiger Peer-to-Peer-Systeme:

- Dezentrale Ressourcennutzung
- Selbstorganisierend
- Peers können alle Funktionen (Client/Server/Router/”Gateway” oder Relay) beliebig annehmen
- Unabhängigkeit vom physikalischen Netz, eigene Topologie
- Mobilität der Peers und somit der Diensteanbieter

Diese Paradigmen decken sich mit den Anforderungen, die an das in der vorliegenden Arbeit beschriebene System gestellt werden.

2.5.2 Entwicklung der Peer-to-Peer-Systeme

Von ersten schnell entwickelten Ansätzen, die sich auf Flooding-Mechnismen stützen (zum Beispiel für File Sharing-Applikationen), welche auch als unstrukturierte Peer-to-Peer-Netze bezeichnet werden, entwickelte die Forschung schnell Mechanismen wie die Distributed-Hash-Tables (DHT's). Mit diesem Ansatz lassen sich verlässliche skalierbare Systeme realisieren, die heute als „strukturierte“ Peer-to-Peer-Netze bezeichnet werden.

Die heute wichtigsten Vertreter von Peer-to-Peer-Basissystemen sind SUN's JXTA [66] und Microsoft's .NET Plattformen [40]. Eine ausführliche Übersicht über die Entwicklung von Peer-to-Peer-Systemen findet sich in [38] oder [60].

2.5.3 Grundmechanismen heutiger Peer-to-Peer Systeme

Die anfänglich in Peer-to-Peer-Netzen verwendeten Flooding[15]-Verfahren zur Informationsverteilung zwischen den Peers haben sehr schnell gezeigt, dass solche Systeme nicht „stabil“ und skalierbar genug sind, um den Anforderungen einer neuen Generation von Internet-Diensten gerecht zu werden. Da trotz der bereits erwähnten Nachteile von Peer-to-Peer-Systemen, die Popularität und Nachfrage nach Peer-to-Peer-Systemen ständig stieg, wurde die Forschergemeinde auf die Problematik aufmerksam. Sie entwickelte neue Verfahren zur dezentralen inhaltsadressierbaren Verwaltung von Daten. Diese Entwicklung ist heute eine der wesentlichsten Komponenten der meisten modernen Peer-to-Peer-Systeme. Eine gute Darstellung der Entwicklung der Peer-to-Peer-Systeme findet sich in [35].

Die beachtenswerteste Entwicklung in diesem Zusammenhang sind, wie beschrieben, die sogenannten verteilten Hash-Tabellen, kurz DHT (Distributed Hash Table). Sie kommen heute in den unterschiedlichsten Ausprägungen in den verschiedenen Peer-to-Peer-Systemen zum Einsatz. Die verschiedenen Realisierungen von DHTs, ihre Eigenschaften und Anwendungsbereiche sollen im folgenden Abschnitt kurz betrachtet wer-

[15] Ein Verfahren zur Informationsverteilung in Peer-to-Peer-Systemen, welches die Informationen unstrukturiert an alle Knoten verteilt

den.

Informationsverteilung in Peer-to-Peer Systemen durch Distributed-Hash-Tables

Bei der Anwendung von DHT's in Peer-to-Peer-Netzen wird das Paradigma, dass alle Knoten in einem System grundsätzlich zu jedem Zeitpunkt die gleichen Eigenschaften haben, zugunsten der Skalierbarkeit des Systems, zum Teil aufgegeben. Es wird eine dynamische hierarchische Struktur eingeführt, innerhalb derer es zwei Klassen von Knoten gibt. Eine Klasse von Knoten, in der Hierarchie weiter oben stehend, übernimmt dynamisch die Verwaltung der im System befindlichen Daten und bildet die DHT. Die restlichen Knoten stellen diesen Dienst dann nicht aktiv bereit, sondern nutzen ihn. Um damit nicht die Flexibilität eines Flooding-basierten Systems zu verlieren, können diese Rollen jedoch nach Bedarf getauscht werden. Die zum Verbund der DHT gehörigen Peer-Knoten werden im allgemeinen auch als Super-Peers bezeichnet. Sollen die bei einem solchen Super-Peer gespeicherten Daten bei dessen Verlassen des Verbundes nicht verloren gehen, müssen entsprechende Managementmechanismen und -protokolle eingeführt werden, um dieses zu verhindern. Dieser Management-Overhead unterscheidet sich nun bei unterschiedlichen DHT-Implementierungen sehr stark und die Skalierbarkeit des Systems hängt wesentlich von diesem ab. Kapitel 6.4.1 geht auf die entstehenden Aufwände des im Rahmen der vorliegenden Arbeit verwendeten Mechanismus ein.

Im folgenden Abschnitt werden verschiedene DHT-Implementierungen vorgestellt und der der vorliegenden Arbeit zugrunde liegende SRDI-Mechanismus behandelt. Es wird ebenfalls auf dessen Eigenschaften gerade in Bezug auf Dynamik und Management-Overhead eingegangen.

Ausprägungen von Distributed Hash-Tables

Die Mechanismen der verteilten Hash-Tabellen stellen den wesentlichen Anteil an der Hintergrundlast eines modernen Peer-to-Peer-Systems. Die jeweiligen Ausprägungen haben unterschiedliche Bandbreiteanforderungen und differierende Leistungsmerkmale

in den Bereichen Skalierbarkeit, Geschwindigkeit der Suchanfrage-Beantwortung und Stabilität gegenüber Knotenfluktuation[16] und Netzeinflüssen. Eine gute Übersicht über die gebräuchlichsten Mechanismen findet sich in [10].

Auf Grund der hohen Zahl an existierenden DHT-Mechanismen wie Chord [11], Pastry [51], Tapestry [64], Bamboo [2] und vielen anderen wird im folgenden nur auf dem in der vorliegenden Arbeit untersuchten SRDI-Mechanismus[17] der JXTA-Plattform eingegangen. Gegenüberstellungen und Bewertungen der verschiedenen Algorithmen finden sich zum Beispiel in [34, 21, 4] und den dort aufgeführten Referenzen.

Der in JXTA eingesetzte SRDI-Mechanismus ist eine DHT-Implementierung, die einen grundlegend anderen Ansatz als die oben erwähnten verfolgt. Hierbei werden die Informationen über Inhalte und deren Ablageposition unsortiert und verteilt gespeichert. Das Wissen über die Position der Inhalte wird hierbei bei *allen* Super-Peers gehalten. Die Verteilung sowie die Suche der Informationen basiert hierbei auf einem heuristischen Ansatz, welcher jedoch nicht, wie in Flooding-basierten Verfahren üblich, alle Knoten des Netzes mit einbezieht, sondern sich auf ausgewählte Super-Peer-Nachbarn stützt [67].

Auf P2P-Syteme, welche sich der oben beschriebenen DHT-Mechanismen bedienen, soll im Folgenden eingegangen werden. Wesentlich an der folgenden Betrachtung ist jeweils das Einsatzszenario und die Skalierbarkeit der Systeme.

2.5.4 P2P-Systeme und ihre Eigenschaften

Zur Nutzbarkeit von Peer-to-Peer-Systemen im Zusammenhang mit E-Learning Anwendungen wird zitiert:

[16] Hinzukommen oder Verlassen beziehungsweise Erreichbarkeit vs. Nicht-Erreichbarkeit eines Knotens
[17] SRDI- Shared Resources Distributed Index

> *In Abwesenheit des „Semantic Web“ sucht man nach alternativen Lösungen, um lokale semantische Netze zu erstellen. Peer-to-Peer-Netzwerke können verwendet werden, um auf Ontologie-strukturierten Repositorien Suchabfragen durchführen zu können. Es kann behauptet werden, dass Peer-to-Peer-Netze schneller performant und effektiv werden können, als das allumfassende „Semantic Web“ entstehen kann. Sichere Nutznießer von solchen semantisch angereicherten Peer-to-Peer-Netzen wären Universitäten, die sich durch Austausch der Lerninhalte gegenseitig bereichern würden. (Aus Web: KOM TUD “Veröffentlichungen Cornelia Seeberg (2003)“)*

Die Klassifizierung der Mechanismen wird anhand der folgenden Kriterien vorgenommen:

Kriterium 1: Suchmechanismen und Speichermechanismen einiger P2P-Systeme

Gnutella: Das Gnutella-System sucht innerhalb seines „Netzes“, das heißt auf den teilnehmenden Peers, „blind“ mit einem Flooding-Mechanismus. Auf diese Art und Weise wird die Wahrscheinlichkeit maximiert, dass ein Datum, das sich auf einem der Peers befindet, ohne zentrales Wissen und Strukturen nach unbestimmter Zeit gefunden wird.

Dieser Ansatz ist sehr bandbreiteintensiv und die Bearbeitungszeit für eine Anfrage kann sehr hoch sein. Betrachtungen zu diesen Verfahren finden sich zum Beispiel in [28].

Napster: Andere Systeme, wie Napster umgehen diese Bandbreiten- und Reaktionszeitprobleme, indem ein hybrides System genutzt wird. Napster verwendet hierbei eine zentrale Serverkomponente, welche Listen aller Dateien und Peers verwaltet. Der Datenaustausch findet jedoch auch bei Napster zwischen den Peer-Knoten direkt statt.

Dieser Ansatz ist zwar wesentlich weniger bandbreitenintensiv, jedoch wirft die zentrale Komponente Probleme auf. Zusätzlich kann, auch bei diesem System, der Zugriff auf ein Datum nur dann erfolgen, wenn der Peer, der das Datum speichert, online ist.

Freenet: Diese Problematik wird in Freenet dadurch umgangen, dass bei diesem System eine gemeinsame Verwendung des verfügbaren Speicherplatzes angestrebt wird.

Hierbei „wissen“ die einzelnen Peers nicht mehr, welche Daten bei ihnen abgelegt sind. Die Datenablage wird hierbei verschlüsselt durchgeführt, um die Modifikation oder das Kopieren der Daten zu verhindern. Zusätzlich wird ein Datum auf mehr als einem Server abgelegt, so dass die Wahrscheinlichkeit für einen erfolgreichen Zugriff auf das Datum nicht mehr von der Verfügbarkeit eines einzelnen Peers abhängig ist. Die Problematik des zentralen Servers bleibt jedoch erhalten.

Kriterium 2: Zuverlässigkeit von Peer-to-Peer-Netzen

Durch die hochdynamische Struktur von Peer-to-Peer-Netzen ist jedoch eine sichere Antwort auf eine Anfrage nicht zu gewährleisten. Gründe dafür liegen zum Einen in der möglichen Überlastung des gesamten Netzes oder darin, dass bestimmte Peers temporär nicht verfügbar sind. Mechanismen, um diese Probleme zu vermeiden, sind die Replikation von Daten und die Nutzung von Strukturierungsmechanismen in P2P-Netzen. Die Zuverlässigkeit wird jedoch noch für längere Zeit unter der des heutigen Internets liegen und es wird auch in absehbarer Zeit nicht möglich sein, allgemeingültige Performance-Garantien in P2P-Netzen zu geben.

Kriterium 3: Peer-to-Peer-Systeme mit E-Learning-Hintergrund

Die im folgenden aufgelisteten P2P-Systeme befassen sich mit Teilfragestellungen aus dem in der vorliegenden Arbeit betrachteten Umfeld. Bei Edutella [15] wird die Verteilung von Metadaten betrachtet, wogegen sich das System SON [12] mit der ontologischen Strukturierung von Overlay-Networks und SWAP mit dem verteilten Wissensmanagement beschäftigen. In den folgenden Abschnitten sollen diese Systeme und ihre Eigenschaften betrachtet und die Arbeiten in die vorliegende Ausführung eingeordnet werden.

Edutella: Das Projekt EDUTELLA basiert auf JXTA und ist der Versuch, eine RDF-basierte Metadaten-Infrastruktur auf dessen Basismechanismen zu spezifizieren und zu

implementieren. Zentrale Fragestellungen sind im Rahmen des Projektes die Implementierung eines Such-Dienstes zum Auffinden RDF-basierter Metadaten, eines Replikationsdienstes zur nachhaltigen Bereitstellung dieser Daten, eines Übersetzungsdienstes, welcher die Nutzung der Daten in anderen Zusammenhängen erlaubt, sowie eines Annotationsdienstes, welcher es erlaubt, Anmerkungen zu beliebigen im System abgelegten Daten zu speichern. Das angestrebte Ziel diese Projektes ist es, diese Metadaten zwischen den verschiedensten JXTA-basierten Anwendungen austauschen und nutzen zu können.

Die erste angestrebte Applikation ist eine Plattform zum Metadatenaustausch zwischen deutschen Universitäten. Weitere Applikationen sind im PROLEARN Virtual Competence Center (VCC) in Planung.

SON: Semantic Overlay Networks
Semantic Overlay Networks beschreiben die Verwaltung von durch ihre Semantik in Gruppen geordnete Informationen innerhalb eines Peer-to-Peer-Verbundes. Die Vorteile einer solchen Einordnung von Inhalten zeigen sich besonders bei der Suche [13]. SON's sind in vielen Bereichen realisiert und beschrieben worden. So bedient sich auch das als nächstes beschriebene System eines SON.

SWAP: Semantic Web and Peer-to-Peer
Das Projekt SWAP verknüpft die Technologien des Semantic-Web und des Peer-to-Peer-Computing, um einen möglichst effektiven, strukturierten Zugriff auf das sich auf den verschiedensten Systemen befindlichen Wissen zu ermöglichen. Es zeigt auf, wie sich die semantische Information des Semantic-Web mit der Leistungsfähigkeit des Peer-to-Peer-Computing verknüpfen lässt und welche Vorteile und Gewinne sich daraus ergeben.

Skalierbarkeitsprobleme und Untersuchungen

Neuere Untersuchungen zur Problematik der Skalierbarkeit von Peer-to-Peer-Netzen zeigen, dass die Hauptproblematik hier an einer neuralgischen Stelle liegt. Der Aufwand an Managementdaten ist dann besonders hoch, wenn es eine hohe Fluktuation

an Daten und Teilnehmern im Netz gibt. Um die Informationen und Teilnehmerdaten aktuell zu halten, steigt die nötige zwischen der Knoten zu übertragende Datenmenge, um dieses zu gewährleisten, stark an. An Strategien die Menge dieser, durch zum Beispiel Update-Meldungen entstehenden, Daten auch und gerade bei sehr dynamischen Peer-to-Peer-Netzen gering zu halten, wird in der Forschung aktiv gearbeitet. Besonderes Augenmerk liegt hier auf der Prämisse, dass die Servicequalität dabei nicht negativ beeinflusst wird. Siehe auch [58].

2.6 Abgeleitetes Anforderungsprofil an ein Server-System

Das aufgespannte Feld der Grundlagen zeigt die Komplexität der Aufgabe, ein Server-System zur Unterstützung von E-Learning-Anwendungen zu spezifizieren. Harte und weiche Randbedingungen spannen ein Feld von unterschiedlichsten Aspekten und Fragen auf, die durch das System betrachtet und beantwortet werden müssen.

Betrachtet man das aufgespannte Feld der verschiedenen Grundlagen dieses Kapitels, so stellt sich die Komplexität des Anforderungsprofils deutlich dar. So ergibt sich aus dem Themenbereich der technischen Grundlagen des E-Learning und der Bereiche Modularisierung und Beschreibung von Inhalten und den darin aufgeführten Anforderungen die zu erfüllenden Erwartungen an das System im Bereich der Inhaltserstellung, -beschreibung, -verwaltung und -suche. Die weiteren Themen der netz- und systemtechnischen Grundlagen, der Sicherheitsproblematik sowie die Entwicklungen und Probleme im Bereich des WWW zeigen darüber hinaus die kommunikationstechnischen Anforderungen, denen das System begegnen muss, um die Randbedingung der einfachen Administration und Installation zu erfüllen. Die Peer-to-Peer-Technologie stellt die zur Lösung eben dieses Problems notwendigen technischen Grundlagen auf. Um die über die Verwaltung und Erstellung der Inhalte hinausgehenden Anforderungen der verschiedensten E-Learning-Anwendungen, wie den Zugriff auf beliebige verteilte Ressourcen, zu realisieren, wird im Rahmen der Arbeit dann noch auf Technologien aus der Architektur von verteilten Systemen, wie Remote Procedure Calls und verteilte Filesysteme,

zurückgegriffen.

Es ergibt sich somit ein Anforderungsprofil, das auf Basis aller dargestellten Grundlagen und Anforderungen die folgenden Punkte erfüllen muss:

- Einfache aber wirkungsvolle Erstellung, Verwaltung, Verwendung und Suche von Inhalten
- Einfache Installation, Administration und Verwaltung des Systems
- Anbindung verschiedener CMS- oder E-Learning-Anwendungen
- Einfache Inhaltserstellung und -konversion
- Unterstützung von Abrechnungsmöglichkeiten
- Verteiltes statusbehaftetes Caching zur Netzlastreduktion, Zugriffszeitreduktion und Sicherung der Verfügbarkeit der Inhalte des Systems
- Unterstützung von Nutzer- und Anwendungsprofilen
- Eine eigene Systemkommunikationsschicht, welche das physikalische Transportnetz abstrahiert und deren funktionale Möglichkeiten systemintern ohne die Einschränkungen des physikalischen Netzes bereitstellt.
- Unterstützung verschiedener anwendungstransparenter Kommunikationsmöglichkeiten über die System-Kommunikationsschicht

Die folgenden Kapitel beschreiben, ausgehend von diesem Anforderungsprofil, ein System, welches im Rahmen der gegebenen Möglichkeiten die aufgezeigten Probleme löst.

Kapitel 3

Grundkonzepte einer verteilten Server-Architektur für E-Learning

Nachdem nun die wesentlichen Grundlagen der Arbeit dargestellt sind, soll nun eine Übersicht über die Einsatzszenarien und Randbedingungen des Systems gegeben werden. Im Rahmen dieses Kapitels steht die Definition der Umgebung, der getroffenen Annahmen sowie der geplanten Einsatz- und organisatorischen Randbedingungen im Vordergrund. Diese Betrachtung stellt dann auch die Basis, auf welcher das System definiert und abgestimmt sowie untersucht wurde, dar.

Für das hier vorgeschlagene System wurde eine deutschlandweite Verteilung der Knoten angenommen. Hierbei soll es allen Nutzern möglich sein, von verschiedenen Orten, das heißt vom Arbeitsplatz, von zu Hause, auf Reisen und in Fremd-Institutionen, ihre Arbeiten und Aufgaben wahrnehmen zu können. Hierbei müssen bei der Architektur des Systems die Randbedingungen der jeweiligen Umgebung berücksichtigt werden. Insbesondere sind hier die Sicherheitsrandbedingungen zu betrachten, welche sich auf die Konnektivität der Anwendungen des Nutzers maßgeblich auswirken.

3.1 Spezifikation der Systemnutzer

Die Nutzer des hier vorgestellten Systems können in vier grundlegende Rollen eingeteilt werden, welche im Folgenden kurz beschrieben werden. Die einzelnen Rollen sind hierbei durch die Aufgaben definiert und Nutzer können verschiedene Rollen gleichzeitig einnehmen. So ist zum Beispiel ein Lehrender oft auch gleichzeitig Autor.

Autor: Ein Autor im System ist für die Erstellung von Lehrinhalten verantwortlich und benötigt dazu schreibenden und lesenden Zugriff auf die Inhaltsressourcen des Systems. Für die Pflege und Weiterentwicklung der Materialien ist zudem noch ein „Rückkopplungssystem“ sinnvoll, auf welches er zumindest lesenden Zugriff benötigt.

Lehrer: Ein Nutzer, der die Rolle des Lehrenden innehat, benötigt lesenden Zugriff auf die Lernmaterialressourcen und sollte zudem in der Lage sein, Anmerkungen an die Materialien anbringen zu können, sowie dem Autor Rückmeldungen, zum Beispiel mit Änderungswünschen, zu geben. Darüber hinaus erstellt er vielleicht Vorlesungsmitschnitte, welche er den Lehrmaterialien zuordnen möchte. Im Falle einer stärker interaktiven Vorlesung benötigt er darüber hinaus eine Möglichkeit zur Kommunikation mit seinen Studenten oder Lernenden, um auf deren innerhalb des Systems gemachten Anmerkungen oder gestellten Fragen während und/oder nach der Vorlesung einzugehen.

Lernender: Die Rolle der Lernenden benötigt ebenfalls eine große Menge an Interaktionsmöglichkeiten mit dem System. Hierunter fallen das Betrachten von Inhalten, das Hinterlegen von Anmerkungen, das Nutzen von „Virtuellen Laboratorien“, die Gruppenarbeit mit anderen Studenten sowie die Kommunikation mit den Lehrenden oder den ihnen zugeordneten Betreuern (Siehe Problematik der „Rückkopplung“ oben).

Administrator: Die Administratoren und Betreiber eines DLE-System-Knotens unterscheiden sich von den bereits erwähnten Nutzergruppen, gerade in ihren Aufgaben, wesentlich. Sie sind nicht eigentlich Nutzer des Systems, sondern sind vielmehr mit dessen Betrieb und Wartung beschäftigt. Auch diese sehr wichtigen Aufgaben, wie das Nutzer- und Rechtemanagement oder die Abrechnung, müssen, ähnlich der bereits beschriebenen Interaktionsmöglichkeiten mit dem System, transparent und mobil wahrzunehmen

sein. Im Rahmen der Sicherheit stellen sie jedoch besondere Ansprüche.

Die Aufgaben, welche zu den einzelnen Rollen gehören, lassen sich auf verschiedene Systemanwendungen oder Systemfunktionen abbilden. Je nach Rolle des Nutzers, werden diese dann in verschiedenen Ausprägungen bereitgestellt und können angewendet werden. Abb. 3.1 gibt diese Rollenabbildung im Groben wieder.

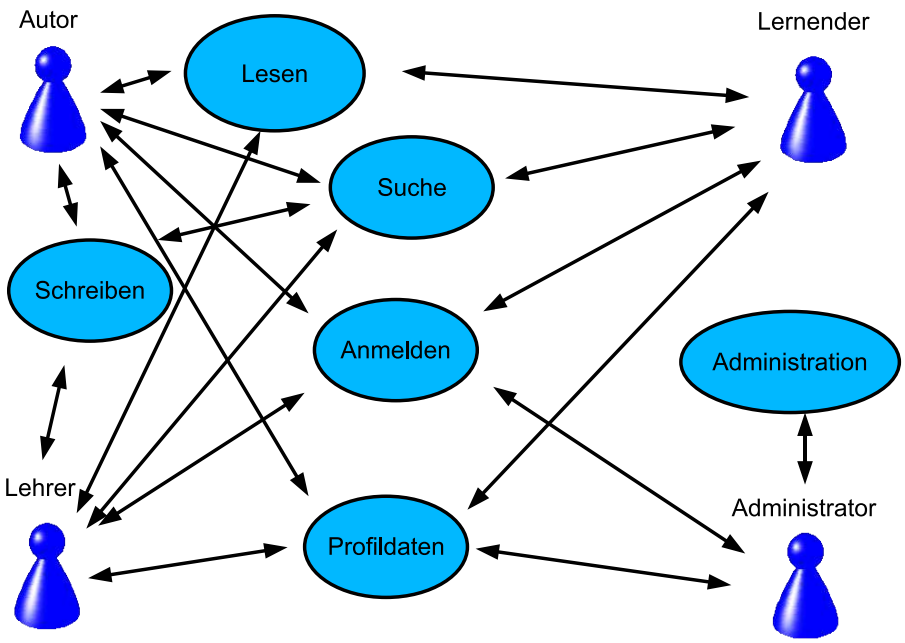


Abbildung 3.1: Nutzerrollen und Systemaufgaben

3.2 Szenarien der Nutzung

Die Nutzer bewegen sich wie beschrieben im Rahmen ihrer Rollen, also nicht an einen Arbeitsplatz gebunden, sind damit mobil. Unter Mobilität der Nutzer wird die zeitabhängige Variation des Aufenthaltsortes des Nutzers verstanden. So können alle Nutzerarten ihre Aufgaben bzw. Arbeiten lokationsinvariant durchführen, ohne dabei wesentliche Beschränkungen bei der Funktionalität der Dienste oder bei der Personalisierung und bei Sicherheitsanforderungen hinnehmen zu müssen.

Die Anwendungsszenarien der einzelnen Nutzergruppen sind hierbei jedoch wesentlich unterschiedlich und stark differierende Anforderungen an das System müssen erfüllt werden. Die folgenden Abschnitte geben einen Überblick über die hier angenommen Basisszenarien.

Die Szenarien sind hierbei:

- Lehrer
  - unterwegs
  - am Heimatinstitut
  - am Fremdinstitut
- Autor
  - am Heimatinstitut
  - am Fremdinstitut
  - unterwegs
- Lernender
  - am Heimatinstitut
  - am Fremdinstitut
  - unterwegs beziehungsweise daheim

Es ist deutlich, dass nicht alle Szenarien in gleicher Häufigkeit vorkommen werden. Die Häufigkeit hängt hierbei maßgeblich von der Rolle des Nutzers ab, so wird zum Beispiel das Szenario eines Autors an einem Fremdinstitut seltener auftreten als das eines Lehrers oder Lernenden an einem fremden Institut. Die unterschiedlichen Häufigkeiten des Auftretens haben hierbei jedoch nur einen geringen Einfluss auf das entworfene System, da die jeweils in Anspruch genommenen Dienste prinzipiell ähnliche oder gleiche Basismechanismen verwenden. Einen durchaus nicht zu vernachlässigenden Einfluss haben sie jedoch auf die Anbindung der rollenspezifischen Anwendungen an die Plattform. Diese Untersuchung bleibt im Rahmen der Arbeit unbehandelt.

3.3 Äußere Rahmenbedingungen

In diesem Abschnitt soll eine Einordnung verschiedener Randbedingungen gegeben werden, welche nicht in die anderen Kategorien hineinpassen. Zum Einen sind das „zwischenmenschliche“, beziehungsweise aus (gefühlten) Abhängigkeiten heraus entstehende Randbedingungen, zum Anderen aus der Auslegung von Erlassen und/oder Richtlinien der Institutionen abgeleitete Randbedingungen, welche einen nicht vernachlässigbaren Einfluss auf das Handeln der jeweiligen Personen haben.

Hier sind zum Beispiel die bereits einleitend genannten sozialen Kontakte der Lernenden untereinander wie Zusammenarbeit oder gemeinsame Erfahrungen in Laboren und Vorlesungen zu nennen. Diese sozialen Interaktionen müssen, gerade weil sie nicht vollständig durch ein solches System ersetzt werden können, eine wesentliche systemische Unterstützung erfahren. Diese wirkt sich auf die durch das System bereitzustellenden Interaktions- und Kommunikationsmöglichkeiten aus.

Eine aus dem Bereich der Richtlinien und Erlasse abgeleitete Randbedingung ist zum Beispiel die Verfügbarkeit von Vorlesungsmaterialien. Da das hier vorgeschlagene System eine breite Verteilung und Wiederverwendung von Lehrmaterialien gerade auch aus anderen Instituten vorsieht, muss sichergestellt werden, dass die „verlinkten“ Materialien auch wirklich zu jeder Zeit dem Lernenden zur Verfügung stehen. Dieses wirkt sich auf die Zugriffs- und Caching-Logik des Systems aus.

Neben diesen Faktoren muss jedoch auch dem im Folgenden betrachteten Aspekt Rechnung getragen werden.

3.4 Wirtschaftliche Anforderungen

Der Aufwand, der in die Lehrinhalte investiert wird, muss von den einzelnen Instituten und Institutionen entweder durch personelle oder finanzielle Ressourcen getragen werden. Um eine Refinanzierung dieser Aufwände zu ermöglichen, wird es in absehbarer

Zeit nötig werden, diese Inhalte zu vermarkten. Die Inhalte müssen somit entweder direkt verkauft werden oder das Institut beziehungsweise die Institution muss sich deren Verwendung angemessen bezahlen lassen. Um dies zu ermöglichen, muss ein Mechanismus bereitgestellt werden, der die dafür notwendigen Funktionalitäten bietet. Die sichere und effiziente Bereitstellung eines solchen Mechanismus ist für die Tragfähigkeit und nachhaltige Nutzung moderner LMS[1]-Systeme, und somit, im Besonderen für die nachhaltige Nutzung des E-Learnings (siehe Kapitel 1) nötig. Eine notwendige Folgerung aus dieser Annahme ist, dass eine AAA- oder A4C-Umgebung in das System integriert sein muss, welche die Aufgabe die wertvollen Inhalte einer breiten Basis an Anwendern in einer sich monetär verwertbaren Weise bereitzustellen, erfüllt.

3.5 Funktionale Anforderungen

Die funktionalen Anforderungen an das System ergeben sich aus der Vielfalt der zu unterstützenden Lehr- und Lernanwendungen. In diesem Abschnitt soll nur eine Zusammenfassung der Anforderungen gegeben werden. Eine genauere Aufstellung der Anwendungen und deren speziellen Anforderungen findet sich in Anhang C. Es ergibt sich aus der dort aufgezeigten Betrachtung, dass neben den bereits oben aufgeführten Anforderungen sich zusätzlich Transport-, Datenzugriffs- und Echtzeitanforderungen aus den im Rahmen des E-Learning zu unterstützenden Anwendungen ableiten. Als Beispielanwendungen für diese erweiterten QoS-Anforderungen an das Transportsystem wären hier Virtuelle Laboratorien oder Konferenzsysteme zu nennen. Die entwickelte Plattform muss zu deren Unterstützung in der Lage sein, diese neuen Anforderungen zusammen mit den bereits genannten restlichen Randbedingungen mit der von ihr bereitgestellten Kommunikationsinfrastruktur zu erfüllen. Diese Anforderung ist jedoch nur dann erfüllbar, wenn entsprechende Mechanismen gerade für die Abrechnung der Nutzung dieser zuletzt genannten Dienste sowie zur dynamischen Anpassung der Topologie der DLE-Knoten an die Kommunikationsanforderungen genutzt werden und das darun-

[1] Learning Management Systeme

terliegende physikalische Transportnetz in der Lage ist, diese Anforderungen zu erfüllen.

Sollten jedoch bereits das darunterliegende Transportnetz oder die angeschlossenen Rechner nicht in der Lage sein, die geforderten Echtzeitkriterien einzuhalten, so kann im Rahmen der Arbeit keine Möglichkeit dieses Fehlen zu umgehen, aufgezeigt werden. Die Anforderungen können jedoch immer auf die Forderung, so wenig Netzlast wie möglich als Overhead zu erzeugen und die Laufzeiten der Pakete im Netz nicht so stark zu verlängern, dass dem Anwender daraus Nachteile entstehen, zurückgeführt werden.

Bevor im Kapitel 5 auf die Architektur des Systems eingegangen wird, soll in Kapitel 4 das entwickelte Werkzeug zur Erstellung von E-Learning-Materialien vorgestellt werden.

3.6 Abgeleitete Grundkonzepte für das System

Betrachtet man die in den vorherigen Abschnitten dieses Kapitels zusammengetragenen Anforderungen an das zu entwerfende System, so lassen sich daraus Grundkonzepte für das System ableiten. Wie bereits aus dem Anforderungskatalog ersichtlich, erstrecken sich diese Konzeptionen nicht nur über die von dem System angebotenen Dienste, sondern wirken auf alle Bereiche eines solchen komplexen Systems. Im Folgenden sollen die Grundkonzepte für die wesentlichsten Bereiche zusammengestellt werden.

Integriertes oder integrierbares Autorensystem: Ein integriertes oder integrierbares Autorensystem ergibt sich aus der Notwendigkeit der einfachen Nutzbarkeit des Systems gerade und besonders durch die erhöhten Aufwände des Autors durch die geforderte Annotation der Materialien mit Metadaten. Auch die Anforderung nach einem einheitlichen Austauschformat für die Lehrinhalte ist mit diesem Grundkonzept problemlos zu realisieren.

Strukturierte Ablage der Lehrinhalte: Diese Anforderung ergibt sich durch die zu Austausch und Wiederverwendung der Materialien notwendigen Auffindbarkeit derselben. Hierzu gehört dann auch der einfache Zugriff auf die Lehrinhalte.

Einfacher Zugriff auf Lehrinhalte: Im Rahmen eines Distributed Learning Environment

muss der dezentrale „problemlose“ Zugriff auf die darin enthaltenen Materialen sichergestellt werden. Ist dieser zu stark erschwert, so wird die Anforderung nach Austausch und Wiederverwendung nicht zu erfüllen sein. Hierbei darf die Sicherheit und Abrechenbarkeit nicht außer Acht gelassen werden.

Einbindung von Sicherheits- und Abrechnungsmechanismen: Um den aus rechtlichen und wirtschaftlichen Gründen notwendigen Zugriffschutz und die Abrechenbarkeit der Materialen und Berücksichtigung des einfachen Zugriffs auf die Materialien erfüllen zu können, muss eine entsprechende Infrastruktur in das System integriert sein. Ohne diese Integration ist auch das folgende Grundkonzept nicht zu erfüllen.

Transparenter, mobiler „Netzzugang“: Die in der heutigen Zeit gebotenen Möglichkeiten der Informationstechnologie ermöglichen dem Nutzer eine hohe Möbilität und die „freie“ Wahl eines Arbeitsplatzes. So muss das entwickelte System, unter der Berücksichtigung der bereits genannten Konzepte, diese Mobilität unterstützen. Dies darf jedoch nicht ohne Betrachtung des folgenden Konzeptes stattfinden.

Ressourcenschonung: Da sich ein solches System, im besten Falle über eine große Anzahl beteiligter Institutionen erstreckt, muss darauf geachtet werden, dass die Konzeption die anfallende auszutauschende Datenmenge so klein wie möglich hält. Bei einem zu hohen Kommunikationsaufwand des Sytems ist dessen Skalierbarkeit in Frage zu stellen, und die Anforderung und das Grundkonzept nach dem „Einfachen Zugriff auf die Dienste des Systems“ wären in Frage zu stellen.

Einfacher Zugriff auf unterstützende Dienste: Das Grundkonzept nach einem einfachen Zugriff auf die Dienste des Systems gilt hierbei für zwei Bereiche. Zum Einen müssen die Basisdienste des Systems ohne große Hürden nutzbar sein, um eine Teilnahme leicht zu ermöglichen. Zum Anderen muss bei der Konzeption darauf geachtet werden, dass auch weitere benötigte Dienste zukünftig leicht bereitzustellen und zu nutzen sind.

Die in den folgenden Kapiteln entworfene Architektur beachtet diese Grundkonzepte und ihre Auswirkungen beginnend mit der Problematik der Autorensysteme.

Kapitel 4

Struktur und Erstellung von E-Learning-Materialien

Zur Erstellung von E-Learning-Materialien sollte ein Autorensystem herangezogen werden, dessen Einstiegsaufwand für den Autor im Bezug auf dessen Einarbeitung sowie der Nutzung bereits vorhandenen Materials möglichst gering ist. Um dieses zu erreichen, wurde ein Transformationssystem entwickelt und implementiert, welches es ermöglicht, modulare mit Metadaten annotierte Lerninhalte mittels herkömmlicher Autorensysteme zu erstellen.

Das folgende Unterkapitel beschreibt das Vorgehen, die Architektur sowie die auftretenden Probleme und ihre Lösungen.

4.1 Lehrmodulstrukturen und herkömmliche Autorensysteme

Lehrmodule zur Verwendung im Umfeld neuer Medien in der Lehre unter der Maßgabe von Wiederverwendbarkeit und Austauschbarkeit müssen die in Abschnitt 2.1.1 aufgezeigten Randbedingungen einhalten. Diese Randbedingungen erfordern eine semantisch getriebene Strukturierung der Daten, so dass die nachfolgenden Mechanismen zur

Speicherung, Suche, Referenzierung und Einordnung der einzelnen Abschnitte respektive (Sub-)Module einer mit einem herkömmlichen Autorensystem erstellten Lehreinheit möglich wird. Dies steht jedoch im Gegensatz zu dem Faktum, dass eine Textverarbeitung im Regelfall nicht semantisch, sondern Layout-getrieben ist. Ein Kapitelanfang zeichnet sich somit nicht durch seine Eigenschaft aus, dass ein Kapitel an dieser Stelle beginnt, sondern dadurch, dass es zum Beispiel in einer größeren Schrift gesetzt wird. Es muss hier ein Weg gefunden werden, die bestehenden Eigenschaften herkömmlicher Autorensysteme zu nutzen, um semantische Strukturierungen zu beschreiben ohne weitgehende beziehungsweise tiefgreifende Änderungen an den Autorensystemen durchführen zu müssen. Ein häufig gewählter Ansatz hierzu bildet die Nutzung von Formatvorlagen wie er in Abschnitt 4.2 für dieses Projekt beschrieben wird. Dieser Ansatz wird auch in den Projekten ITO, MBDB, METACOON, FIGARO und XDiML verfolgt [47, 39, 43, 26]. Jedoch werden bei der späteren Nachverarbeitung der vom Autorensystem erstellten Quelldokumente weitergehende Probleme offensichtlich, die in den genannten Projekten nicht vollständig gelöst wurden. Die Probleme und die in der vorliegenden Arbeit gefundene tragfähige Lösung wird in Abschnitt 4.5 beschrieben. Die hier erarbeitete und beschriebene Lösung wurde in das Projekt MetaCoon [39] integriert und wird dort weiterentwickelt.

4.2 Formatvorlagen zur automatisierten Transformation und Metadaten-Erstellung

Um die automatisierte Modularisierung und die Metadaten-Erstellung zu ermöglichen, muss der Autor die Materialien mittels einer vorgegebenen Formatvorlage erstellen, beziehungsweise bereits vorhandene Inhalte mit dieser Formatvorlage neu formatieren. Die hier vorgeschlagene Formatvorlage stellt die minimal notwendigen Absatzvorlagen sowie Zeichentypen zur Erstellung von Lehrmaterialien zur Verfügung.

Der in Abb. 4.1 dargestellte Satz an Zeichen und Absatzformaten orientiert sich an den

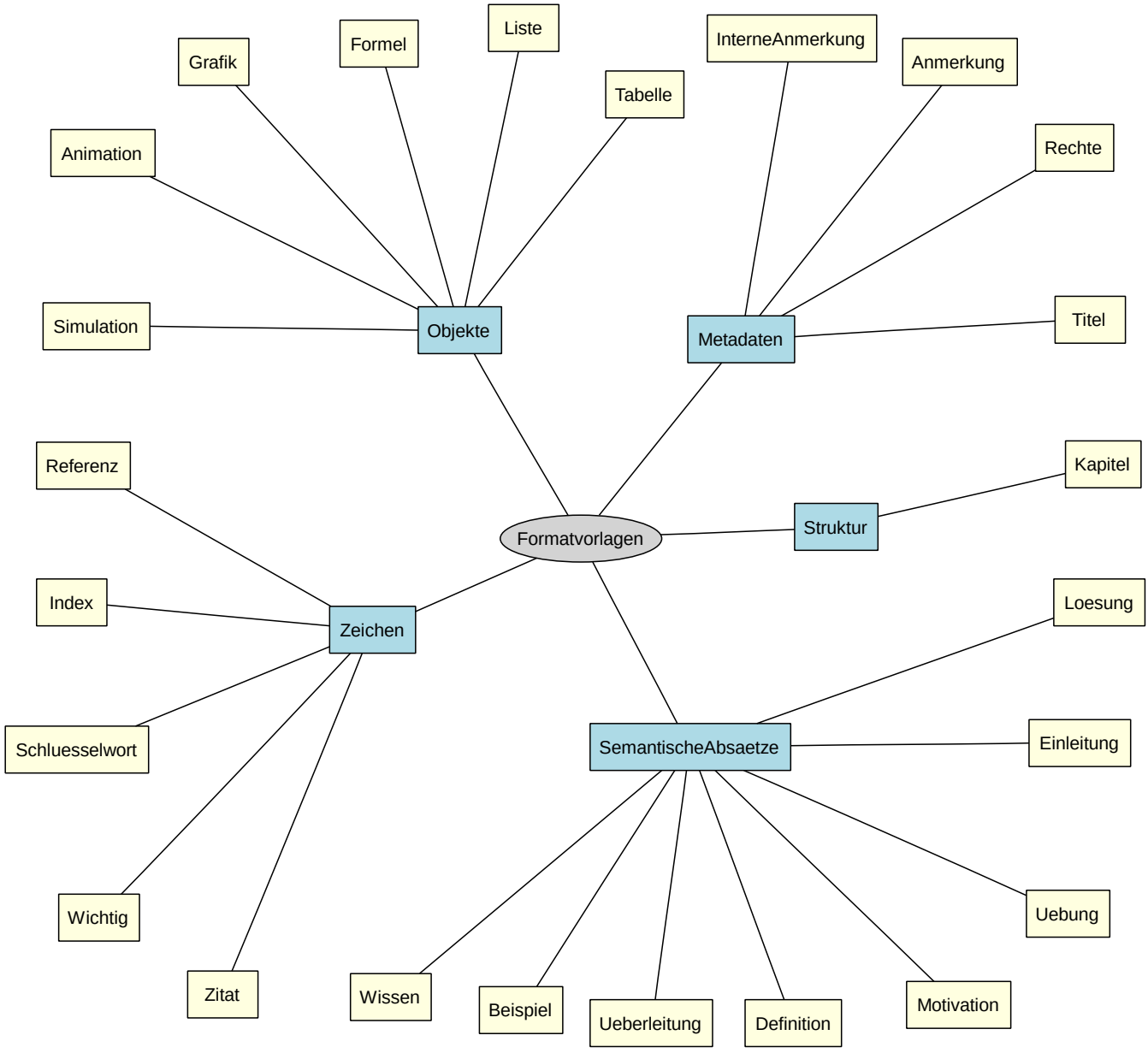


Abbildung 4.1: Formatvorlagen zum Erstellen von Lehrmodulen

Anforderungen für Materialien in der Informationstechnologie und Informatik und unterteilt die definierten Formatvorlagen in fünf Gruppen mit jeweils speziellen Anforderungen. Die fünf Gruppen sind:

- Objekt-Formatvorlagen dienen zur Beschreibung von eingebetteten Objekten wie Grafiken, Formeln, Bilder und dergleichen.
- Strukturierungs-Formatvorlagen dienen zur Unterteilung des Dokuments in Abschnitte.
- Semantik-Formatvorlagen dienen zur Annotation von Teilen eines Abschnittes um deren Inhalte zu klassifizieren.
- Zeichen-Formatvorlagen dienen zur speziellen Formatierung und Annotierung von Zeichenketten innerhalb eines semantischen Absatzes.
- Metadaten-Formatvorlagen dienen zur Beschreibung von notwendigen Metadaten für einen der oben genannten Fälle.

Diese fünf Gruppen finden sich so auch im Zwischenformat, siehe Abschnitt 4.6.2, wieder. Die Formatvorlagen sind hierbei so gewählt, dass sie den Autor so wenig wie möglich einschränken, leicht verständlich, übersichtlich und leicht anwendbar sind. Trotz dieser Kriterien erfüllt er die Randbedingungen aus Abschnitt 4.1 und die auf diese Art erstellten Lehrmaterialien enthalten genügend Semantik, um eine Modularisierung im Sinne der in Abschnitt 2.1.2 vorgestellten Anforderungen automatisiert vorzunehmen. Die hier vorgestellte Lösung wurde, durch das ITO-Projekt angeregt, entwickelt. Sie wurde dort und im Rahmen der „Leipziger Informatiktage“ vorgestellt und wird heute in anderen Projekten (unter anderem MetaCoon) weiterentwickelt und eingesetzt [20].

Ein Problem, das sich für den Autor bei der Anwendung der beschriebenen Formatvorlagen stellt, ist die semantische Richtigkeit der Abfolge der im Text eingesetzten Formatvorlagen.

4.3 Erweiterungen im Autorensystem zur Nutzerunterstützung

Zur Unterstützung des Autors wird eine Erweiterung des Autorensystems vorgeschlagen, welche nur minimal in die Funktionen des Autorensystems eingreift, dem Autor jedoch genügend Hilfsmittel an die Hand gibt, die Formatvorlagen zu verwenden. Das heißt, der Autor wird durch die zusätzlichen Funktionen bei der korrekten Anwendung der Formatvorlagen und besonders bei komplizierten Vorgängen, welche zum Beispiel eine bestimmte Abfolge von Absatz-Formaten verlangen, unterstützt, indem die notwendigen Parameter in Dialogen abgefragt und Formate sowie Daten in der korrekten Abfolge in das Dokument eingefügt werden. Ein Beispiel dieser Zusatzmenüs und Assistenten findet sich in Abb. 4.2. Abb. 4.3 zeigt die dazu notwendige, geschlossene Werkzeugkette. Es werden die Schritte der Erstellung der Materialien, der Transformation in das Zwischenformat und die Ausgabe sowie die für den Autor wichtige Anzeige von Fehlern und Anmerkungen einbezogen. Die technische Beschreibung der Transformation findet sich in Abb. 4.6 und Abschnitt 4.5. Zur Erstellung von wiederverwendbaren und austauschbaren Materialien kommt nun noch die Erstellung und Anheftung zusätzlicher Informationen hinzu. Eine Lösung dazu wird im folgenden Abschnitt aufgezeigt.

4.4 Einbettung von Metadaten

Die in dem erstellten Material enthaltenen Module müssen, um den oben beschriebenen Anforderungen gerecht zu werden, noch mit weiteren Metadaten angereichert werden. Diese lassen sich, auch bei Einsatz der oben beschriebenen Formatvorlagen, nicht, beziehungsweise nicht ohne weiteres, automatisch aus der semantischen Struktur der Absätze und deren Formaten ableiten. Es entsteht daraus eine Notwendigkeit, diese weitergehenden Informationen zusätzlich zu den Inhalten, Formaten und Strukturierungen im erstellten Dokument abzulegen. Aus verschiedenen Gründen sollten diese Informationen jedoch nicht als „normaler“ Text im Absatz erscheinen. Es muss deswegen ei-

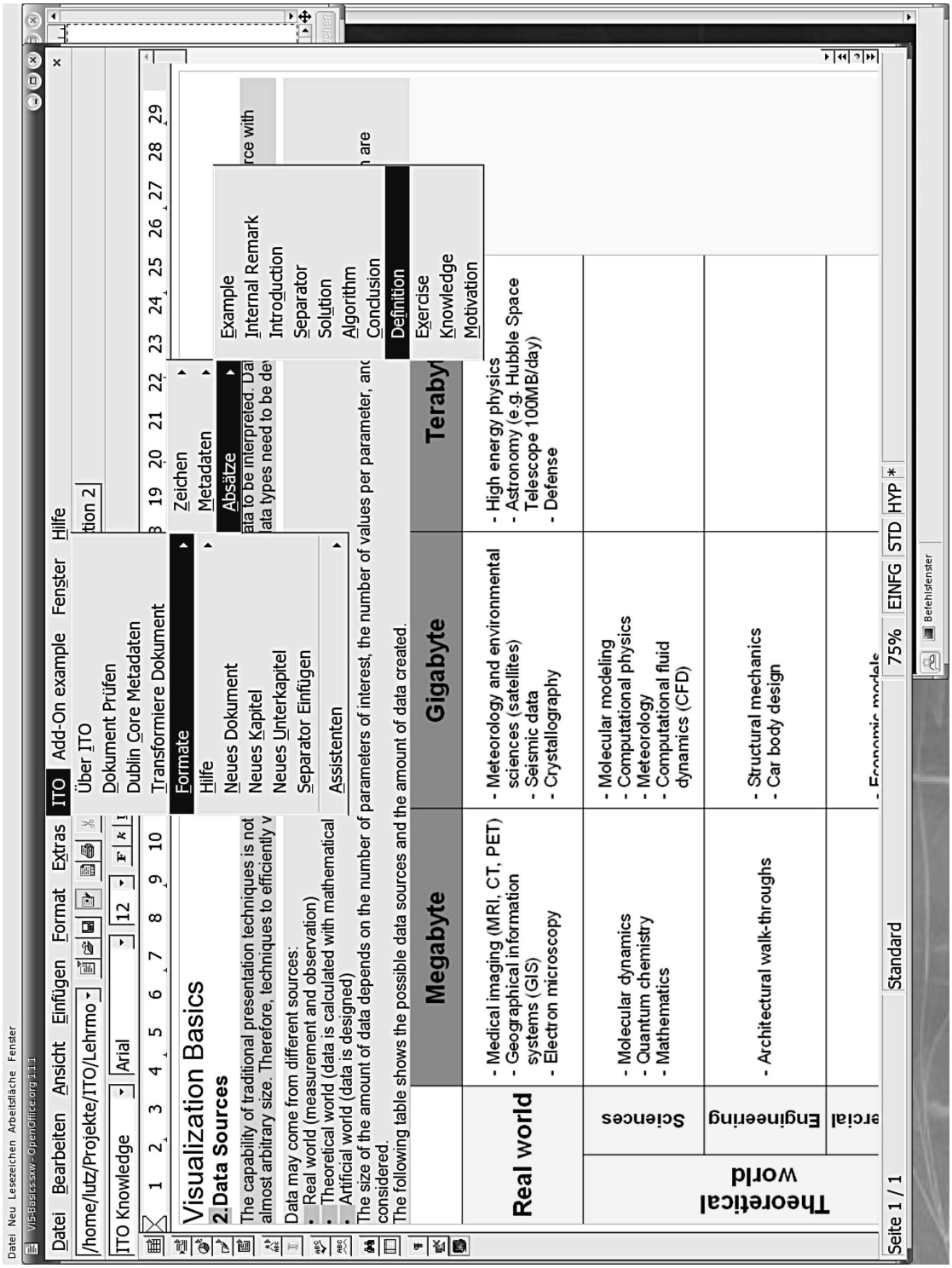


Abbildung 4.2: OpenOffice.org als Autorensystem

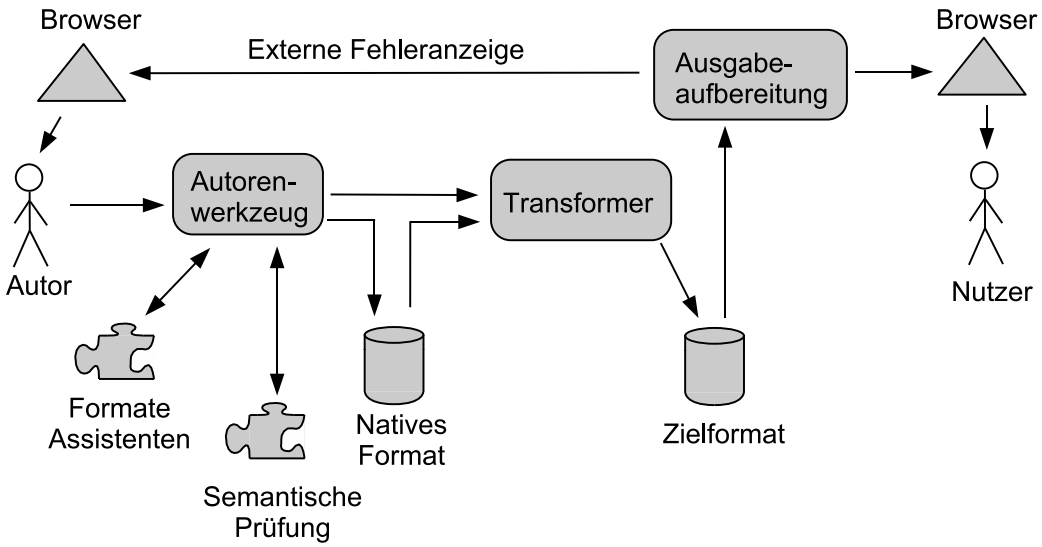


Abbildung 4.3: Werkzeugkette für den Autor

ne Möglichkeit gefunden werden, diese Zusatzinformationen für den Autor „transparent“ abzulegen. Eine notwendige Randbedingung ist außerdem, dass diese Daten auch bei einem Wechsel des nativen Autorenwerkzeuges erhalten bleiben. Es muss somit ein Mechanismus gewählt werden, der werkzeugübergreifend existent ist. Dieser Mechanismus sollte darüber hinaus so generisch sein, dass die gespeicherten Daten nicht konvertiert werden müssen.

Nach Betrachtung der in den verschiedenen Autorensystemen zur Verfügung stehenden Mechanismen, erscheint hierbei das Anheften von Notizen, respektive Kommentaren an beliebige Textstellen, der am besten zu diesem Zweck nutzbare Mechanismus zu sein. Dieser wird dann auch, wie im folgenden beschrieben, für diesen Zweck genutzt.

In allen betrachteten Systemen erlaubt der Kommentar- respektive Notizmechanismus, an beliebigen Stellen im Text Informationen zu hinterlegen. Diese Informationen können hierbei auch strukturierte Dokumente sein. Für das Hinterlegen von Metadaten zu den einzelnen Absätzen wird folgendes Vorgehen definiert:

- Die Metadaten zu einem Absatz werden immer am Beginn der ersten Zeile desselben hinterlegt, um das Auffinden der Kommentare zu erleichtern
- Die Kommentare enthalten in der ersten Zeile des Kommentars eine leicht erkenn-

bare Signatur, so dass die Metadatenkommentare von anderen zu unterscheiden sind

- Die im Kommentar hinterlegten Metadaten werden in einer DC-, LOM- oder SCORM-konformen XML-Struktur hinterlegt und so eine nachhaltige Nutzbarkeit erreicht.

Für den Autor werden, um dieses Vorgehen zu unterstützen, Zusatzfunktionen geschaffen, die das Anlegen und Ändern dieser Metadaten erlauben. Die Abb. 4.4 zeigt die Annotationsmöglichkeiten und einen der Dialoge für den Autor. Um den aufwändigen Prozess der Annotation der Lernmodule mit Metadaten weiter zu vereinfachen, wird zusätzlich eine hierarchische Vererbung von Metadaten über die Absätze respektive Lernmodule hinweg vorgeschlagen und in den Grundzügen implementiert. Die Vererbung von Metadaten stellt hierbei eine interessante Aufgabe dar, da die Vererbungsrichtung der Informationen erst durch die Eigenschaften des betrachteten Merkmals festgelegt wird, und nicht von vornherein über alle Merkmale hinweg bestimmbar ist. Beispiele für unterschiedliche Vererbungsrichtungen sind Autoren- und Schlüsselwort-Felder. Die Information über den Autor eines Absatzes kann aus dem Vorgängerkapitel ermittelt werden, dagegen muss jedoch die Information der Schlüsselworte zum Vorgängerkapitel hin weitergegeben werden. Die Implementierung verzichtet auf diese Unterscheidungsmechanismen und betrachten vorerst nur die Vererbung der Autoreninformationen.

4.5 Transformation in Module und Metadaten

Die Speicherung der Inhalte erfolgt in allen herkömmlichen Autorensystemen, wie beschrieben, immer Layout-getrieben. Es gibt bei der Speicherung keine oder kaum hierarchische Strukturen, welche für die Transformation der Materialien in Module und die dazugehörigen Metadaten, herangezogenen werden können. Um diese Transformation mit Hilfe der in Abschnitt 4.2 vorgestellten Formatvorlagen durchzuführen, wird ein komplexer Analyse- und Strukturierungsprozess vorgeschlagen und implementiert. Abb. 4.5 stellt die einzelnen Stufen dieses Prozesses grob dar.

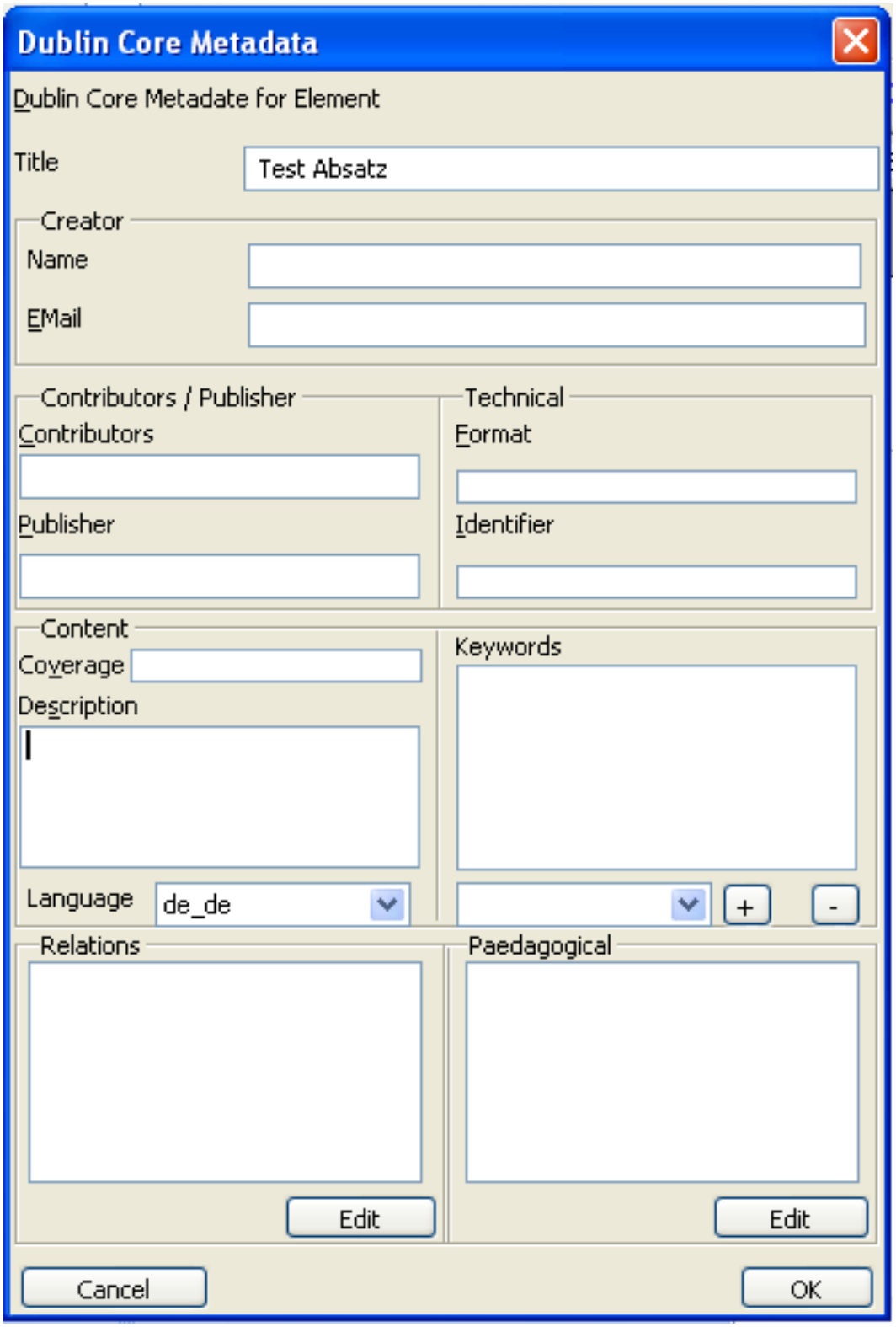


Abbildung 4.4: Metadaten Annotations Dialog

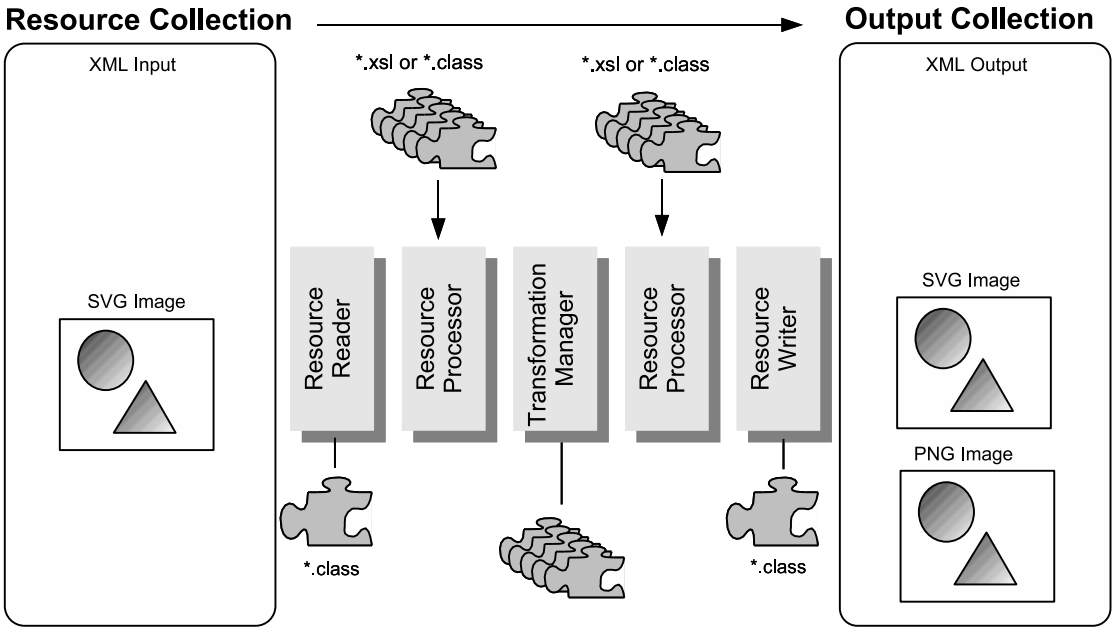


Abbildung 4.5: Der Transformationsprozess

Um die Nutzung des Transformationsprozesses für die Anwender möglichst einfach und transparent zu gestalten, wird das jeweilige Autorensystem, im Falle des Prototyps ist das OpenOffice, erweitert und angepasst. Die Erweiterungen und Anpassungen sehen erweiterte Menüs und Funktionen vor, mit denen der Anwender die Erstellung und Transformation seiner Inhalte durchführen kann, ohne im Vorfeld tiefgehende Schulungen besucht haben zu müssen (siehe auch Abschnitt 4.3).

Der eigentliche Transformationsprozess ist in Abb. 4.6 dargestellt und läuft wie folgt ab:

Um das Layout-getriebene schwach strukturierte mit Hilfe der Formatvorlagen und eines herkömmlichen Autorensystems erstellte Originaldokument in das stark strukturierte und Semantik-getriebene Zwischenformat konvertieren zu können, wird der im Folgenden dargestellte, mehrstufige Transformationsprozess definiert. Der Prozess gliedert sich grob in drei wesentliche aufeinanderfolgende Teile. Zuerst wird das Originaldokument untersucht und dabei die darin referenzierten externen Materialien in eine temporäre Ressourcenablage, einem virtuellen Filesystem, abgelegt. Bei der darauffolgenden Transformation wird der Inhalt des Originaldokumentes selbst verarbeitet. Der Inhalt wird analysiert, in seine Bestandteile zerlegt und diese dann, ihrem semantischen

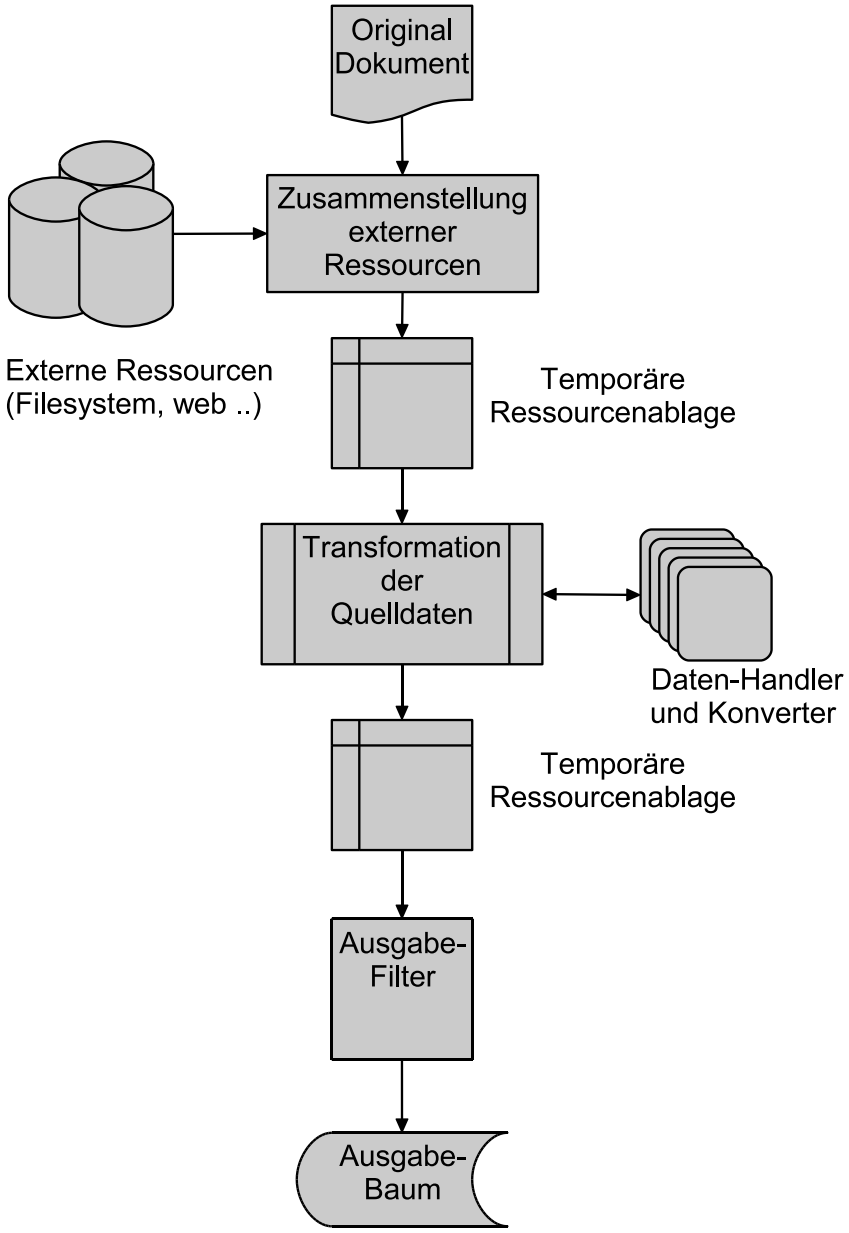


Abbildung 4.6: Ablaufdiagramm des Transformationsprozesses

Kontext entsprechend, in einer Baumstruktur (XML-DOM[1]) abgelegt. In diesem Schritt erfolgt ebenfalls die Generierung und Zuordnung der entsprechenden UID's für die im Inhalt identifizierten Module und „Atome".

Als letzter Schritt wird diese Struktur in einem weiteren virtuellen Filesystem abgelegt, welches entweder in ein lokales Filesystem, oder, zum Beispiel, in ein an das DLE[2] angeschlossenes CMS[3] erfolgen kann.

Den komplexesten Teil dieses Prozesses stellt hierbei die eigentliche Transformation des Inhaltes dar. Der Prozess geht davon aus, dass sich der Inhalt des Originaldokumentes zumindest in erkennbar voneinander abgrenzbare, sequenziell auftretende Bereiche untergliedern lässt. Diese werden von im Vorfeld spezifizierten „Data Handlers and Converters" in die entsprechende hierarchische und semantische Struktur überführt. Hierbei haben diese Prozessoren die Aufgabe, die Ausgabestruktur des Zwischenformates zu erzeugen, indem sie die, mit den Formatvorlagen im Originaldokument erstellten Zusammenhänge der sequenziellen Teile erkennen. Es ergibt sich daraus, dass die Handler oder Prozessoren an das jeweilig genutzte Autorensystem angepasst sein müssen, da die Autorensysteme einen Einfluss auf die Abfolge der identifizierten Teile haben. Die im Rahmen der Arbeit entstandenen Handler (Siehe Tabelle 4.1) decken den Bereich der bereits beschriebenen Formate und Objekte ab. Sie müssen das in Abb. 4.7 dargestellte Interface implementieren und werden im Transformationsprozess wie in Abb. 4.8 gezeigt, eingesetzt.

Die implementierten Handler und Prozessoren sind ausreichend, um verschiedene, mit den Formatvorlagen aus Abschnitt 4.2 erstellten, Dokumente mit Erfolg in das Zwischenformat zu überführen.

Zusätzlich, um den Nutzer bei Fehlanwendung der Formatvorlagen zu helfen, werden beim Transformationsprozess auffallende semantische Fehler, wie falsche Abfolge von Formatvorlagen, fehlende oder falsche Metadaten, Definitionen ohne entsprechendes Stichwort und so weiter, als Fehlermeldungen in das Ausgabeformat eingebettet. Die

[1] XML-DOM: Programmiermodell für XML-Dokumente „Document Object Model"
[2] DLE - Distributet Learning Environment, das in der vorliegenden Arbeit entworfene System
[3] Content Management System

Tabelle 4.1: Element Handler für das Zwischenformat und OpenOffice

Name	**Aufgabe**
AlgorithmHandler	Behandelt Algorithm-Abschnitte
AnimationHandler	Behandelt Animations-Abschnitte
AnnotationHandler	Behandelt Anmerkungs-Abschnitte
ChapterHandler	Behandelt Kapitel
CitationHandler	Behandelt Literaturverweise
ConclusionHandler	Behandelt Folgerungs- o. Aussage-Abschnitte
DefinitionHandler	Behandelt Definitionen
ExampleHandler	Behandelt Beispiel-Abschnitte
ExerciseHandler	Behandelt Übungs-Abschnitte
FormulaHandler	Behandelt Formeln
IllustrationHandler	Behandelt Illustrationen
ImageHandler	Behandelt Bilder
IndexHandler	Behandelt Index-Einträge o. -Verweise
InternalRemarkHandler	Behandelt interne Vermerke des Autors
IntroductionHandler	Behandelt einleitende Abschnitte
KeywordHandler	Behandelt Schlüsselworte
KnowledgeHandler	Behandelt Wissensabschnitte
LineBreakHandler	Spezielle Behandlung von Zeilenumbrüchen
ListHandler	Behandelt Aufzählungen
ListItemHandler	Behandelt einzelne Punkte in Aufzählungen
MotivationHandler	Behandelt Motivations-Abschnitte
ReferenceHandler	Behandelt Referenzen
RightsHandler	Behandelt Rechteabschnitte für Metadaten
SignificantHandler	Behandelt Hervorhebungen
SimulationHandler	Behandelt eingebettete Simulationen
SolutionHandler	Behandelt Lösungsabschnitte für Übungen
StrongSignificantHandler	Behandelt Hervorhebungen
TablesHandler	Behandelt Tabellen
TitleHandler	Behandelt Titel von Dokument und Abschnitten
TransitionHandler	Behandelt Überleitungen

```java
/*
 * ITOElementHandler.java
 *
 * Created on 28. November 2002, 15:54
 */

package ind.fi.ito.transform;

import java.util.ListIterator;
import java.util.logging.Logger;

/** Interface for Element Handlers
 * @author Lutz Finsterle
 */
public interface ITOElementHandler {

    final int ERROR_INFO    = 1;
    final int ERROR_WARNING = 2;
    final int ERROR_ERROR   = 4;
    final int ERROR_PANIC   = 8;

    final String [] ERROR_NAMES = {"","info","warning","","error","","",
                                                              "","panic"};

    public void setContext(Context c);

    /**
     * @param o
     * @return
     */
    public boolean canHandle(Object o);

    /**
     * @param o
     * @param it
     * @return
     */
    public Object constructElementFor(Object o, ListIterator it);

    public boolean validate(Object m, Object o , ListIterator it,
                            boolean insertErrors);

    public Object createError(int type,String message );
}
```

Abbildung 4.7: Java-Implementierung des Handler Interface

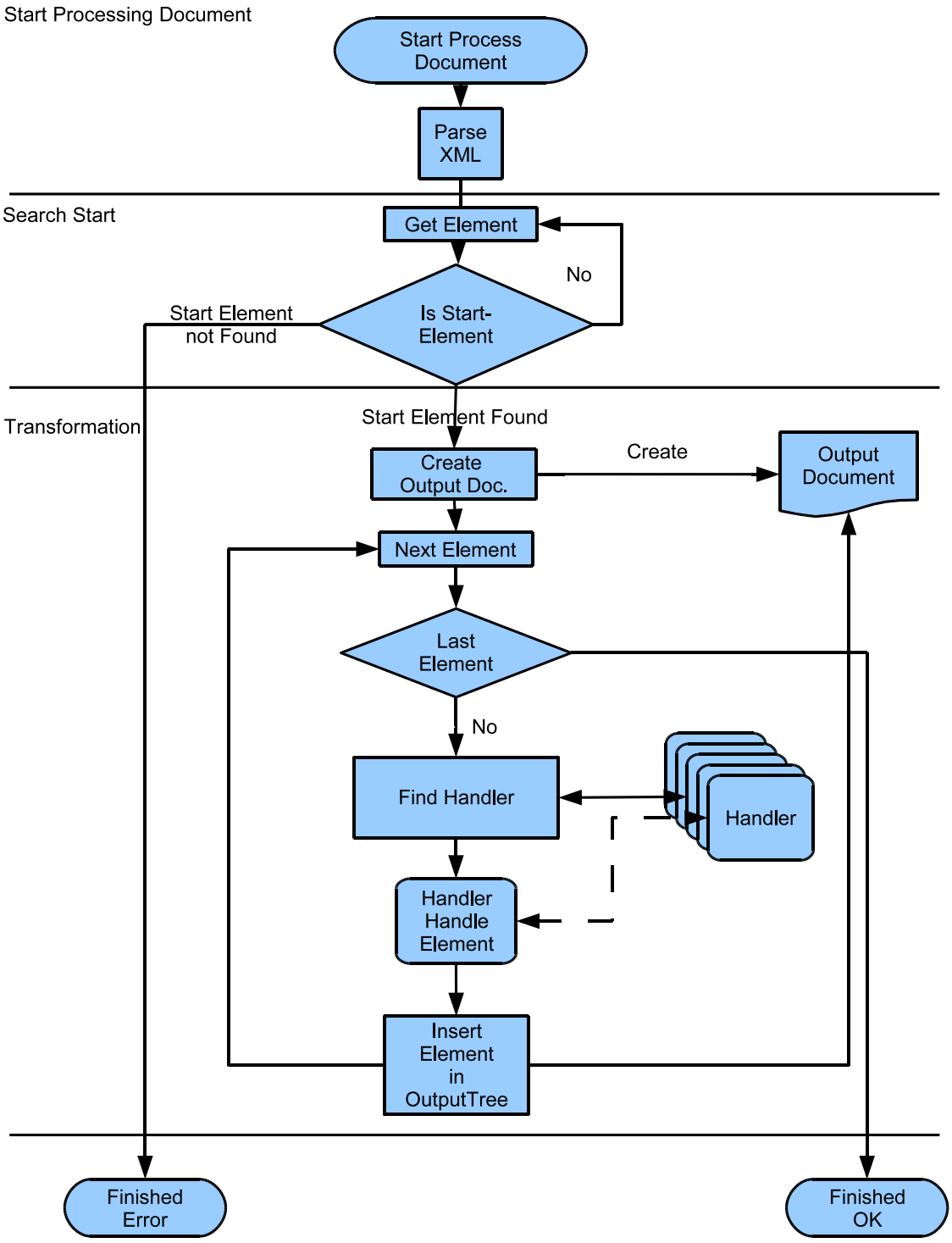


Abbildung 4.8: Ablaufdiagramm der Handler-Nutzung

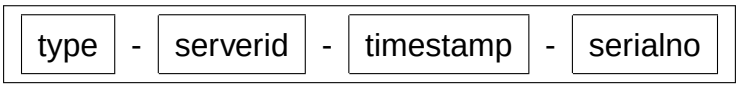
Struktur des Ausgabeformates ist im folgenden Abschnitt beschrieben.

4.6 Das Ausgabeformat

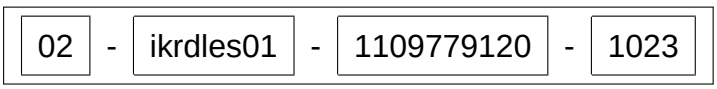
Das Ausgabeformat wurde in der vorliegenden Arbeit in einer Form spezifiziert, die es ermöglicht, dass erstellte Lernmaterial sowohl in einem Stück, als auch in modularisierter Form abzuspeichern. Besonderes Augenmerk lag hierbei auch auf der Eigenschaft, dass das in dieser Form abgelegte Dokument einfach in andere Formate, im besonderen SCORM und LOM, konvertiert werden kann. Zu diesem Zweck wurde ein stark strukturiertes Ausgabeformat in XML entworfen. Das Ausgabeformat ist in XML-Schema [69] spezifiziert und in leicht erweiterbarer Form entwickelt. Für die Spezifikation der Metadaten wurde als Minimum eine Übermenge der Dublin-Core Spezifikation [49] definiert. Ein einfaches Beispiel der Anwendung des Formates findet sich in Abb. 4.9. Die vollständigen Schemadefinitionen finden sich in Anhang D, eine kurze Beschreibung der Schema Struktur findet sich in Abschnitt 4.6.2.

4.6.1 Unified Resource Identifier

Es werden standardisierte Identifizierer zur eindeutigen Identifikation der Module und ihrer Komponenten genutzt. Diese sind so beschaffen, dass sie in dem nachfolgend vorgestellten System eindeutig sind.



zum Beispiel:



Das Schema wurde durch das CANDLE-Konsortium [18] in dieser Weise definiert und ist für die Anwendung in dem in der vorliegenden Arbeit beschriebenen System bestens

```
<doc:document
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:SchemaLocation="http://ITO/Zwischenformat/Document"
    xmlns:char="http://ITO/Zwischenformat/Character"
    xmlns:bib="http://ITO/Zwischenformat/BibTexSupport"
    xmlns:err="http://ITO/Zwischenformat/Error"
    xmlns:doc="http://ITO/Zwischenformat/Document"
    xmlns:obj="http://ITO/Zwischenformat/Object\"
    xmlns:meta="http://ITO/Zwischenformat/Meta"
    >
    <meta:title>Ein Beispiel Kurs</meta:title>
        <doc:chapter cid="2-ikrdles01-12414-14522">
        <meta:title>Einführung</meta:title>
        <doc:motivation pid="2-ikrdles01-12414-12141" >
            <char:text>
                Dies ist die Motivation für das Beispiel
            </char:text>
            <char:text reftype="internal"
                       tid="2-ikrdles01-12414-11231"
                       type="reference">
                Eine Referenz
            </char:text>
            <obj:illustration oid="2-ikrdles01-12414-12141"
                              type="illustration">
                <obj:frame>
                    <obj:bounds height="1" width="1" x="1" y="1" />
                    <obj:image  base="" height="" width=""
                            href="" mime="" name="" />
                </obj:frame>
                <obj:title>Name of the Illustration</obj:title>
                <meta:annotation></meta:annotation>
                <meta:rights>  </meta:rights>
            </obj:illustration>
        </doc:motivation>
    </doc:chapter>
</doc:document>
```

Abbildung 4.9: Struktur des Ausgabeformates

geeignet. Durch die eindeutige Bezeichnung unter Einbeziehung der Serveridentifikation ist ein Zugriff auf eine so referenzierte Ressource aus jedem System-Kontext heraus möglich. Notwendig ist hierzu die Erreichbarkeit des Servers beziehungsweise das Vor-

handensein des Moduls im lokalen Cache und die entsprechende Zugriffsberechtigung auf das Modul (Siehe Abschnitt 2.3.4). Heute werden solche Identifizierer im Allgemeinen nach den Richtlinien des „Digital Object Identifier“(DOI)-Konsortiums gebildet. Die Form entspricht nicht dem hier verwendeten Format sondern setzt sich folgendermaßen zusammen:

<DIR>.<REG> /<DSS> Dabei sind die drei Bestandteile definiert als:

<DIR> Der notwendig anzugebende standardisierte Directory-Code (Numerisch)

<REG> Der notwendig anzugebende fest vergebene Code des registrierten Nutzers (Numerisch)

<DSS> Vom Nutzer frei vergebener Anteil des DOI

Lassen sich die Anwender der hier vorgestellten Plattform bei der Internationalen DOI-Foundation registrieren, so können die im Rahmen der hier vorgestellten DLE-Plattform genutzten UID’s eineindeutig in einen DOI überführt werden. Insofern stellt die Verwendung der in CANDLE spezifizierten UID’s keine Einschränkung der Plattform dar.

4.6.2 Schemadefinition des Zwischenformates

Die Definition des Zwischenformates basiert auf der Dokumentbeschreibungs- und Dokumentdefinitionssprache XML-Schema. Hierbei werden sowohl die Namensraum- und Modularisierungsmöglichkeiten als auch die Typendefinitionseigenschaften dieser Sprache genutzt, um die Definition möglichst leicht anpassbar, wartbar und eindeutig zu gestalten. Die Definition unterscheidet hierzu sechs Gruppen von Elementen und deren spezifische Eigenschaften. Die Gruppen sind:

- Eigenschaften von Zeichenketten
- Dokument- und Absatzeigenschaften

- Eigenschaften eingebetteter Objekte
- Eigenschaften von Metadatenknoten
- Eigenschaften von Literaturverweisen
- sowie die Eigenschaften von eingebetteten Fehlerinformationen

Es wird so erreicht, dass die zu den Gruppen gehörigen spezifischen Eigenschaften in getrennten Dateien gepflegt werden können und so eine sehr übersichtliche Struktur der Formate und seiner Elemente entsteht. Da das Zwischenformat eine rein semantisch orientierte Struktur hat, und bei seiner Erstellung die Formatierungsinformationen verworfen werden, wird im folgenden die Darstellung des Zwischenformates beschrieben, siehe auch [20].

4.6.3 Darstellung von Inhalten im Zwischenformat

Sind die Module, respektive deren Inhalte einmal in das Zwischenformat konvertiert, so lassen sich diese dann leicht in eine beliebige, darstellbare Form überführen. Hierzu müssen mittels Transformationsmechanismen den Inhalten Layout-Informationen hinzugefügt werden, welche während des Transformationsprozesses in das Zwischenformat entfernt wurden. Eine Möglichkeit ist die Transformation mittels XSLT(eXtensible Stylesheet and Transformation Language) und CSS (Cascading Stylesheets) in html, pdf oder ein anderes Format. Ein Beispiel zeigt Abb. 4.10. Die hier dargestellten Eigenschaften und Möglichkeiten der Transformation erfüllen die Anforderungen aus Abschnitt 2.1.1. Darüber hinaus bieten moderne Autorensysteme wie das hier genutzte OpenOffice.org oder die Office-Suite von Microsoft, die Möglichkeit, beliebige XML Dokumente zu importieren und über spezielle Stylesheets zu editieren. Diese Möglichkeit schließt den Kreis eines Round-Trip-Engineerings, da durch diesen Mechnismus das erzeugte Zwischenformat wieder in das mit Formatvorlagen annotierte native Format des Autorenwerkzeugs überführt werden kann.

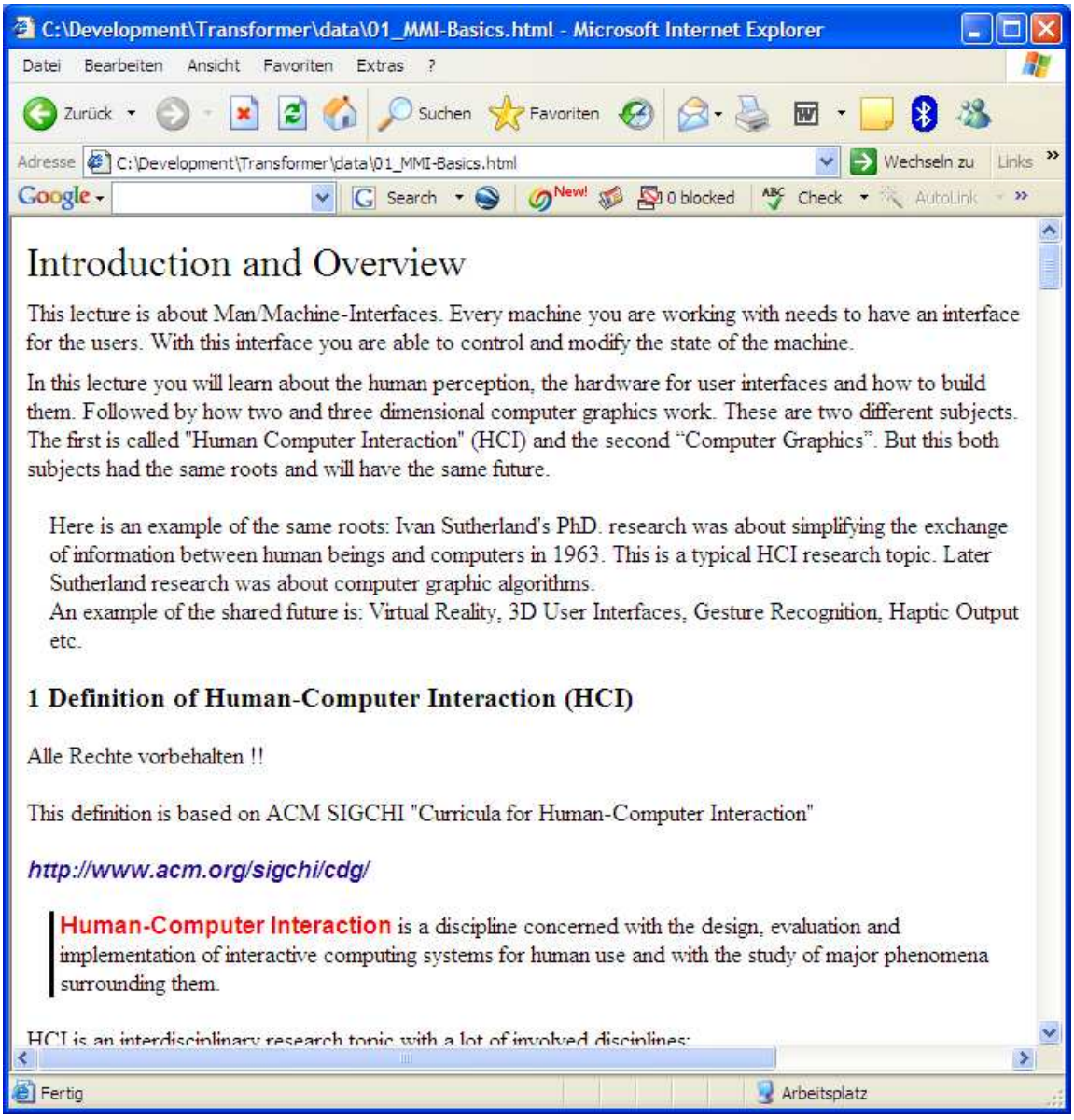


Abbildung 4.10: In HTML-transformiertes Lehrmodul

Kapitel 5

Die Architektur des Server-Systems

Das DLE-Server-System stellt nun, unter Einhaltung der genannten Randbedingungen und Voraussetzungen, eine Plattform für E-Learning-Applikationen zur Verfügung.

Vereinfacht kann die Kommunikationstopologie des DLE-Systems wie in Abb. 5.1 dargestellt aufgefasst werden. Bei der Betrachtung des Systems wird vom eigentlichen Transportnetz und seinen Strukturen abstrahiert. Betrachtet werden nur die entstehenden Verkehrsaufkommen an den Übergängen zwischen einem Server und dem abstrahierten Transportnetz. Diese Annahme darf getroffen werden, da in den Skalierbarkeitsbetrachtungen, die in der vorliegenden Arbeit durchgeführt werden sollen, davon ausgegangen wird, dass die benötigte Bandbreite weit unter der bereitgestellten Bandbreite des Transportnetzes liegt. Vielmehr wird sogar die Skalierbarkeit des Systems in diesem Sinne definiert, das heißt das System wird nur dann als skalierbar bezeichnet, wenn die von ihm im Mittel benötigte Bandbreite wesentlich kleiner ist als diejenige, die heute im Internet zur Verfügung steht.

5.1 Basisstruktur des Kernels

Den Kern der Architektur des Serversystems bildet der Micro-Kernel, in welchem die Verwaltung aller Dienste sowie die Abstraktion der Plattform vom Betriebssystem und

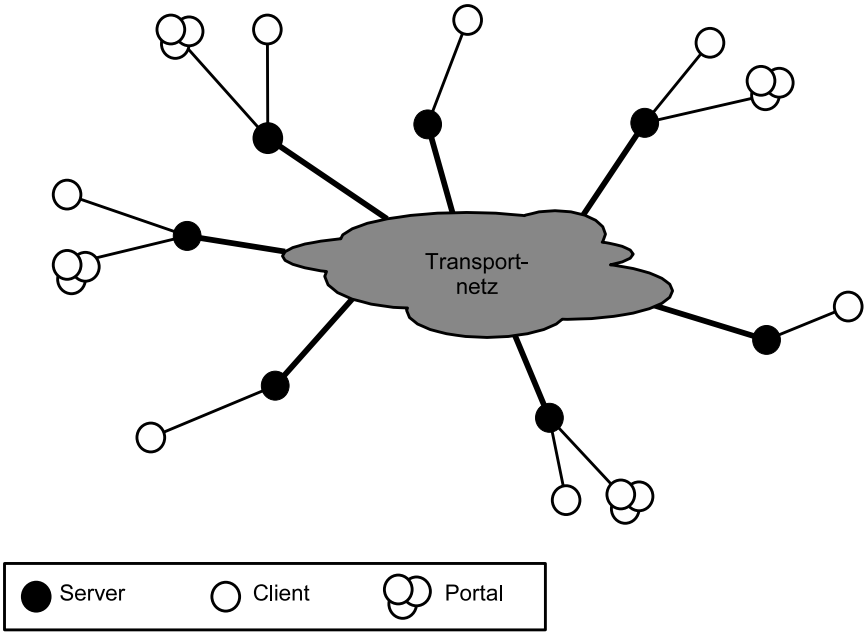


Abbildung 5.1: Übersicht über das DLE-System

vom Transportsystem durchgeführt wird. Die wesentlichen zentralen Dienste sind in Bild 5.2 dargestellt.

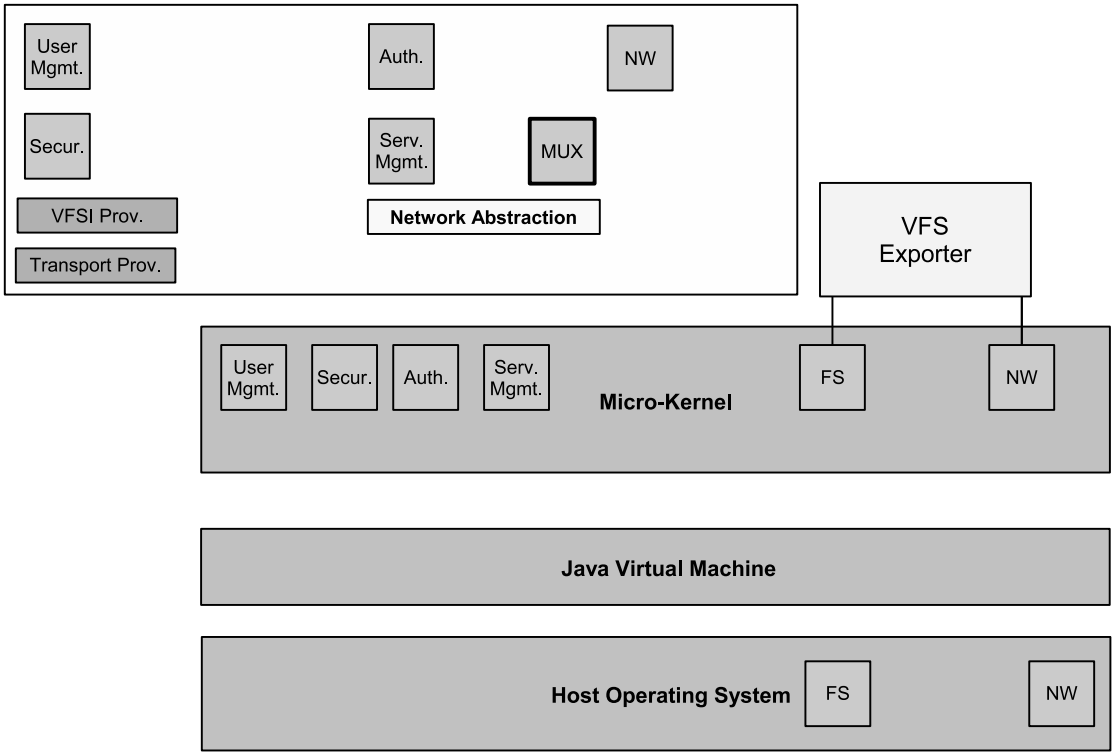


Abbildung 5.2: Basisdienste des Mirco-Kernels

Die dargestellten Dienste bilden die Grundlage, um mit dem in der vorliegenden Arbeit beschriebenen Gesamtsystem die gestellten Anforderungen erfüllen zu können. Diese Basisdienste sind von einander abhängig, da sonst die notwendigen Sicherheitsmechanismen nicht realisiert werden können.

5.2 Hierarchische Struktur und Namensräume des DLE

In Peer-to-Peer-Systemen wird im Allgemeinen nicht zwischen Dienst-Nutzern und Dienst-Anbietern unterschieden. In dem hier vorgeschlagenen System wird zum Teil mit diesem Paradigma gebrochen. Dieses ist notwendig, um sowohl den Kommunikationsaufwand des Peer-to-Peer-Systems zu verringern und um, parallel dazu, die für das AAA-System notwendige Hierarchisierung der Knoten zu erreichen. In Bild 5.3 ist die Gruppenstruktur des DLE gezeigt. Aus den dargestellten Überlegungen und den Strukturen des für den

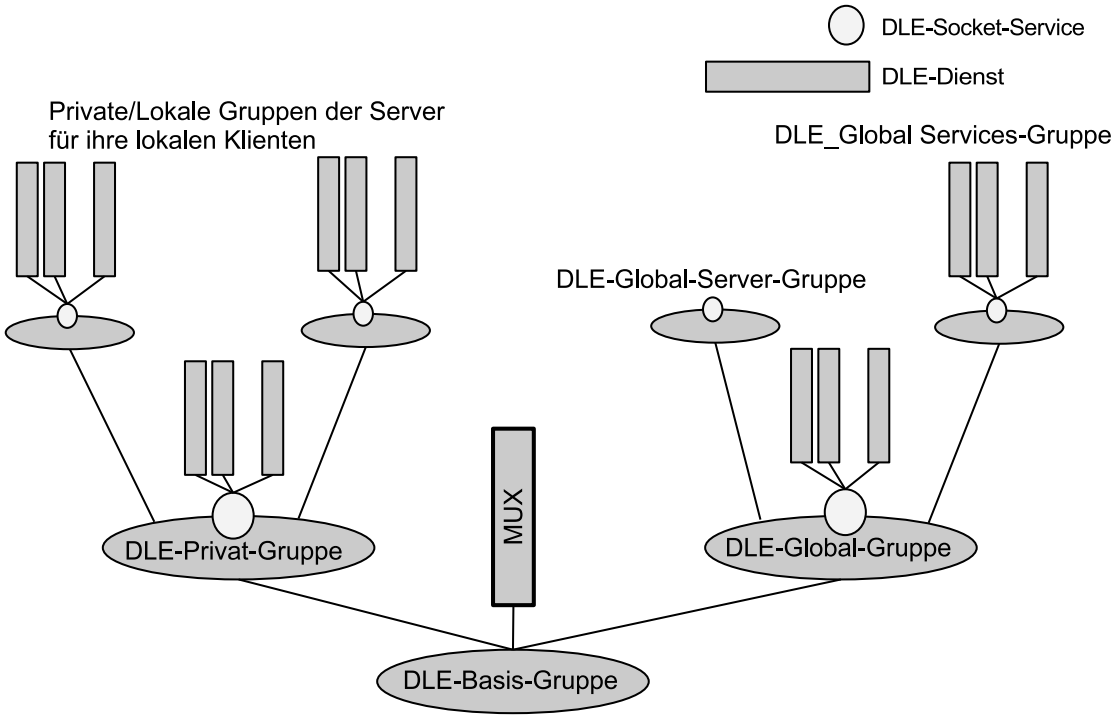


Abbildung 5.3: Die Gruppenstruktur des DLE

Prototypen gewählten Peer-to-Peer Systems leitet sich die in Abb. 5.6 dargestellte und

den DLE-Prototypen zugrunde liegende Gruppenstruktur ab. Hierbei sind die Namensräume und Addressierungsschemata wie folgt gewählt und definiert.

Das Adressierungsschema hat den in Abb. 5.4 dargestellten Aufbau.

```
Finale_Addresse[.  .][  .][  ].BasisAddresse
      ↑                              ↑
   "Port"  "weiter Addressanteile"  "IP-Adresse"
```

Abbildung 5.4: Aufbau der Adresssätze

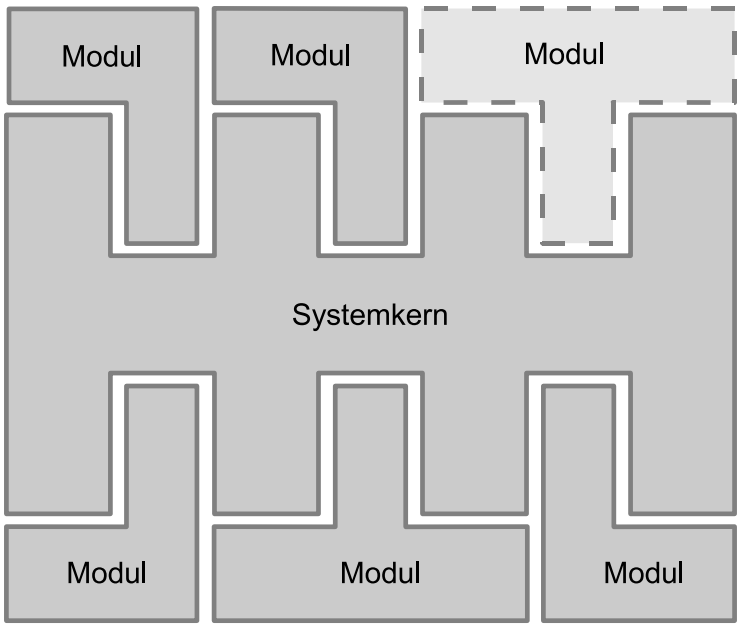


Abbildung 5.5: Abstraktion des DLE-Systems

Die eingebetteten, nicht näher bezeichneten Adressanteile sind entweder die aufgelöste Service- oder Gruppen-ID oder ein lokal aufzulösendes Namenskonstrukt. Hierbei finden alle Verkehrslenkungsaktivitäten anhand der Basis-Adresse im Overlay-Netz statt. Das darauf folgende Multiplexing von der Hauptadresse zu den lokalen Unteradressen wird dann durch den Multiplex-Service der DLE-Basis-Gruppe durchgeführt. Somit kann das DLE-System wie das in Abb. 5.5 vereinfacht dargestellte Schema eines Systemkerns mit verschiedenen direkt daran angegliedernten Diensten betrachtet werden. Die Dienste sind hiernach mit den zentral vom Systemkern bereitgestellten Mechanismen zu Kommunikation, Datenzugriff, User- und Zugriffsmanagement und weiteren (siehe Abb.

5.2) verbunden und können die damit bereitgestellten Funktionen, je nach Notwendigkeit, nutzen.

Die Gruppenstruktur im Prototyp gibt die in Abb. 5.6 dargestellte Struktur wieder.

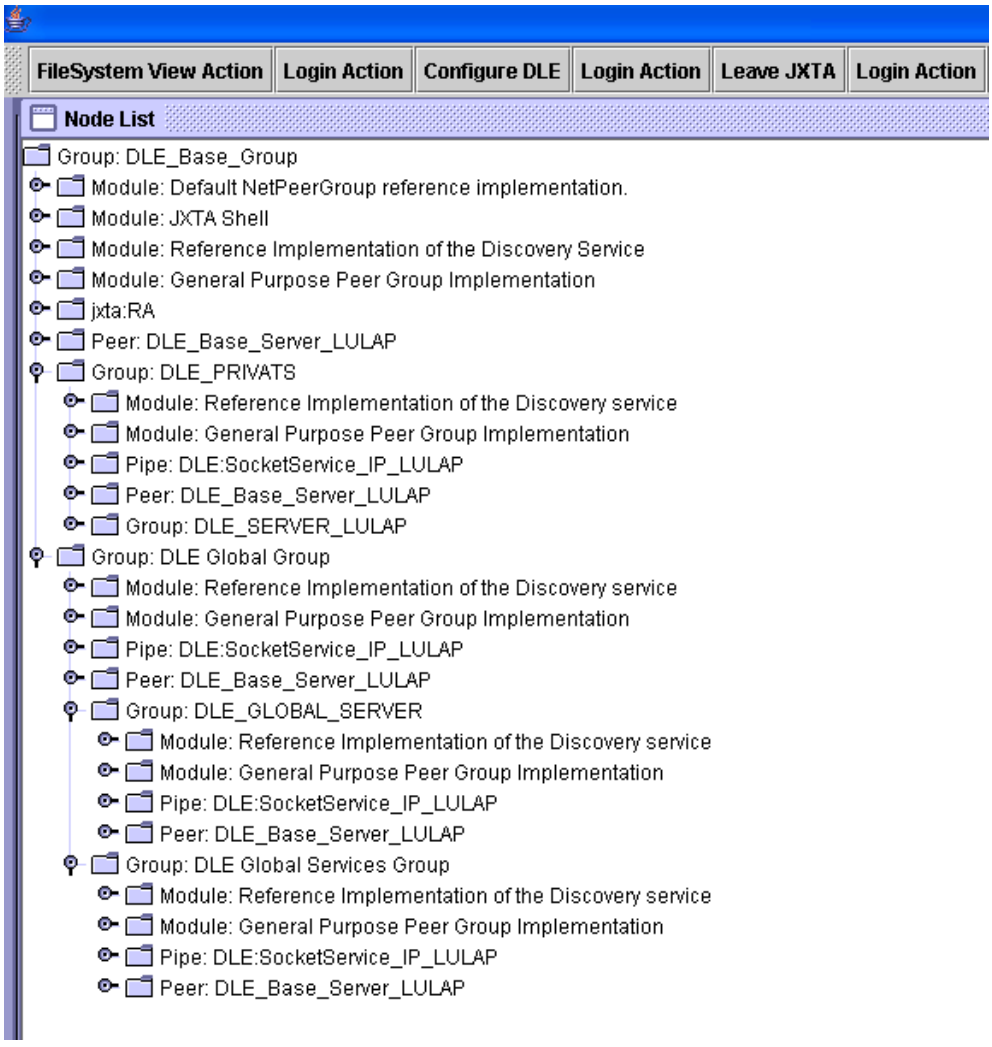


Abbildung 5.6: Gruppenstruktur des Prototypen

5.3 Kommunikationsmechanismen für die DLE-Dienste

Um die Kommunikationsschnittstelle der DLE-Dienste weitestgehend vom Transportnetz unabhängig zu gestalten, wurde eine an Unix-Domain-Sockets angelehnte Schnittstelle zur Kommunikation der Dienste untereinander eingeführt. Ein weiterer Vorteil dieser

Schnittstelle ist, dass nur ein Kanal des unterliegenden Transportnetzes für alle Dienste benötigt wird. Für die Peer-To-Peer-Plattform bedeutet dies, dass die Anzahl der im System zu verteilenden netzlastbestimmenden Advertisements pro Server und Gruppe nicht von der Anzahl der im System verfügbaren Dienste abhängig ist.

5.3.1 Die Transportschicht

Die Transportschnittstelle des Systems muss in der Lage sein, alle bereitgestellten Transportmechanismen hinter einer generalisierten Schnittstelle zu verbergen. Hierzu wird ein modulares Verfahren vorgeschlagen, welches einen transparenten Zugriff auf verschiedene Transportmechanismen ermöglicht. Dieses bietet den „Nutzern“ des Systems die Möglichkeit, alle vom System bereitgestellten Transportdienste auf einfache und gleichartige Art und Weise zu nutzen. Eine leichte Anpassbarkeit des Transportmechanismus an neue Technologien oder andere Transportsysteme ist somit inhärent gegeben. (Siehe auch die STREAM’s-Beschreibung in Kapitel 2.3)

Im Prototyp werden hierbei zwei Transportdienste implementiert. Ein Dienst, der den Zugriff auf die vom Betriebssystem bereitgestellte BSD/Socket-Schnittstelle erlaubt, sowie ein Dienst, der die Kommunikation über das JXTA Peer-to-Peer-System erlaubt. Hierbei werden jeweils zwei Arten der Kommunikation unterstützt. Ein „Nachrichten“-basiertes Verfahren, das den gerichteten, ungesicherten Austausch von Nachrichten erlaubt und insbesondere für RPC-Protokolle mit inhärenter Fehlertoleranz geeignet ist, sowie ein „Datenstrom“-basiertes Verfahren, das besonders zur dauerhaften, fehlergesicherten Kommunikation zweier Dienste, wie zum Beispiel Virtuellen-Laboren, geeignet ist. Die Transportebene unterteilt sich hierbei in mehrere Unterebenen, in denen spezifische Aufgaben erfüllt werden. Bild 5.7 zeigt eine Übersicht über die Schichten.

In der Micro-Kernel Sicherheitsschicht können die zum sicheren Transport der Nachrichten nötigen kryptografischen und andere Maßnahmen als Module in die Nachrichtenverarbeitungskette eingefügt werden. Die Sicherheitsschicht ist hierbei unterhalb der eigentlichen Transportschicht (DLE-UDP ,DLE-TCP)angesiedelt um transparent in die

DLE-Verkehrsströme eingeschaltet werden zu können.

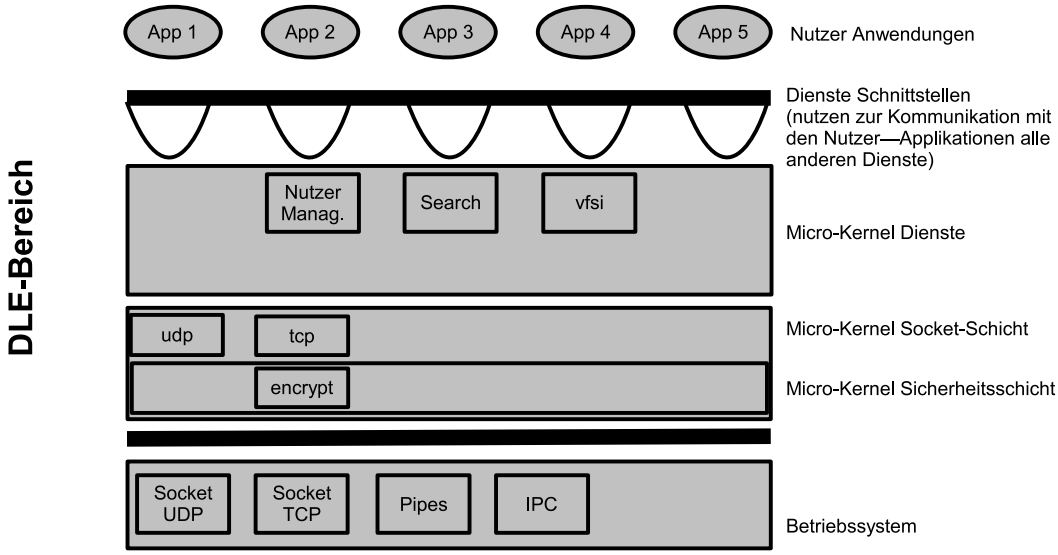


Abbildung 5.7: Der Transportmechanismus

5.3.2 Der DLE-Socket-Service

Der DLE-Socket-Service bietet zwei Arten der Kommunikation für einen DLE-Dienst. Einen Nachrichten-basierten Mechanismus und einen Strom-basierten Mechanismus. Der erste Mechanismus ist hierbei an UDP und der zweite Mechanismus ist an TCP orientiert, ohne jedoch die TCP-inhärente Flusskontrolle zu implementieren. Auf diesen Teil der Implementierung des TCP-Protokolls kann verzichtet werden, da davon ausgegangen wird, dass dieses durch die unteren Schichten der gewählten Kommunikationsmechanismen erfolgt.

Nachrichten-basierte Kommunikation

Bei der „Nachrichten“-basierten Kommunikationsmethode ist ein UDP-ähnlicher Nachrichtentransportdienst ohne Fehlersicherung implementiert worden. Dieser Mechanismus ist für die meisten Fälle ausreichend, in denen eine Sicherungsschicht leicht in die Applikationsschicht integriert werden kann, wie die prototypische Implementierung des RPC-Mechanismus zur Nutzung durch das VFS zeigt.

Strom-basierte Kommunikation

Es wird im Umfeld des E-Learning immer wichtiger, Medienströme zu Vorlesungsmaterialien zur Verfügung zu stellen. Da auch diese Medienströme, wie auch die Vorlesungsunterlagen selbst, den Anforderungen an Abrechenbarkeit unterliegen, muss ein Mechanismus bereitstehen, der die Zugriffe auf solche Inhalte in abrechenbarer Weise zulässt. Die Verteilung dieser Medienströme stellt zusätzlich im Allgemeinen erhöhte Anforderungen an das System, da hier wesentlich auch Qualitätsrandbedingungen (QOS-Anforderungen) eingehalten werden müssen, um eine nutzbare Darstellung der Inhalte zu gewährleisten.

5.4 DLE-Dienste

Auf den beschriebenen Kommunikationskanälen setzen die zur vollständigen Funktion des DLE zur Abdeckung der genannten Randbedingungen und Anforderungen die dazu notwendigen Dienste auf. Im Folgenden sollen die zentralen Dienste der DLE-Plattform vorgestellt, sowie Ihre Funktionalität und Architektur näher erläutert werden.

5.4.1 Nutzer-Management

Die Nutzerdaten und die Nutzer des Systems müssen dezentral verwaltet werden, um die Anforderung nach Eigenständigkeit der teilnehmenden Partnerinstitutionen zu gewährleisten. Jede Institution, die einen Server innerhalb des Systems betreibt, kann somit ihre Nutzer selbst verwalten, oder die Verwaltung an eine übergeordnete Instanz abgeben und dort durchführen lassen. So kann ein Institut einer Universität beispielsweise einen eigenen Server für ihre Inhaltsdaten betreiben, die Verwaltung der Nutzer aber an, zum Beispiel, die Fakultät abgeben.

Um den nach Kapitel 2.3.4 notwendigen Schutz der Nutzer zu gewährleisten, werden Pseudonyme zur Verwendung außerhalb der eigenen Umgebung eingeführt. Diese wer-

den automatisiert von der den Nutzer verwaltenden Institution durch das System erstellt und verwaltet. In der hier entworfenen Architektur wird dieses von den Heimatservern im Zusammenspiel mit den von diesen bereitgestellten Login-Modulen (siehe Kapitel 5.4.2) realisiert.

5.4.2 Login (Authentication) und Pseudonymverwaltung

Der Login-Dienst dient zur sicheren Authentifizierung und Autorisierung von lokalen und mobilen Nutzern. Ein wesentliches Problem ist hier ebenfalls die Sicherheit und der Schutz der Nutzer, das heißt Schutz ihrer Privatsphäre durch Pseudonyme und verifizierte Login-Module. Es muss sichergestellt werden, dass nur Softwaremodule des Heimatservers, dem der Nutzer vertraut, relevante Informationen über den Nutzer erhalten. Zu diesem Zweck wird ein „Login-Managementsystem“ vorgeschlagen, das dieser Anforderung, so weit wie technisch realisierbar, entspricht. Das System ist in [68] aufgearbeitet und dargestellt.

5.4.3 Nutzerprofil-Verwaltung

Um die Forderung nach Individualisierung der Lehrangebote und der Anwendungen zu erfüllen, müssen, je nach Anwendung und Nutzer, eine Menge an Daten bereitgestellt werden. Hierzu wird für das hier entworfene System eine Profilverwaltung vorgeschlagen, welche sich an die Anforderungen der Anwendung anpassen lässt und die notwendigen Profildaten bereitstellen und speichern kann. Diese Bereitstellung kann innerhalb des Systems, für DLE-angepasste beziehungsweise -basierte Anwendungen erfolgen, oder über die VFSI-Schnittstelle nach außen exportiert werden. So wird gewährleistet, dass Nutzer generell in ihrer gewohnten Umgebung arbeiten beziehungsweise lernen können. Sollen die Profildaten des Nutzers nach außen gegeben werden, so muss das System, auf dem sich der Nutzer befindet, entsprechend konfiguriert werden. Dieses kann zum Beispiel durch das Einbinden eines Teiles des von DLE bereitgestellten Filesystems in den Dateibaum des Nutzers oder der Applikation erreicht werden.

Die im System abgelegten Profile und die applikationsspezifischen Daten der Nutzer werden auf dem Heimatserver des Nutzers abgelegt und werden in das nachfolgend dargestellte Schema überführt.

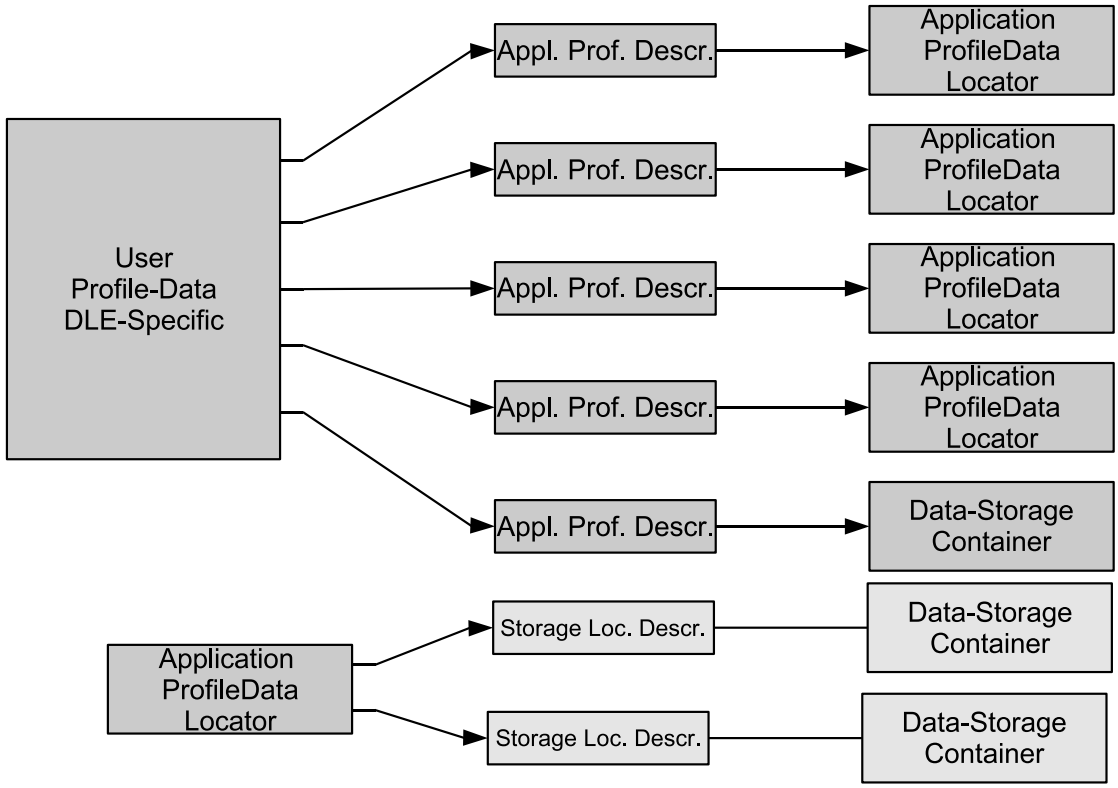


Abbildung 5.8: Schematische Darstellung der Profildatenablage

In der Darstellung in Abb. 5.8 wird die Trennung zwischen der DLE-spezifischen und den anwendungsspezifischen Daten deutlich. Die Trennung ist notwendig, um die mannigfaltigen Anforderungen der verschiedenen im E-Learning-Umfeld eingesetzten Anwendungen, für deren globale oder nutzerspezifischen Voreinstellungen zu erfüllen. Die DLE-spezifischen Daten werden als XML-Dokument abgelegt. Das Format für diese XML-Dokumente ist in Anhang D dargestellt. Die anwendungsspezifischen Daten werden in der Form von Directory-Strukturen abgelegt. Hierbei bekommt jede dieser Strukturen einen zugeordneten Container, welcher dann über das VFSI-System entweder direkt an die Anwendung angebunden oder über das Betriebssystem des Rechners des Users von der Anwendung zugegriffen werden kann. Die hierzu bereitstehenden Mechanismen werden in den folgenden Abschnitten zum DLE-VFS beschrieben.

5.4.4 Suche

Suchalgorithmen aus der Literatur wenden ähnliche Mechanismen wie den im DLE verwendeten Suchmechanismus an. So wird die Abstraktion zwischen Information und dem Ort ihrer Speicherung bereits in [9] beschrieben. Das Projekt Ocean Store [3] betrachtet ebenfalls den Zugriff auf Daten, welche auf verschiedene Rechner abgelegt werden und auf welche dann über den netztechnisch nächsten Rechner im Verbund zugegriffen wird. Das DLE-System erhält dieses Paradigma jedoch nicht für alle Informationsarten aufrecht. So werden die eigentlichen Lehrinhalte nicht generell, sondern nur nutzungsgetrieben auf mehrere Rechner verteilt. Die zum Auffinden der Inhalte notwendigen Metadaten und die verschiedenen über diese gelegten Strukturierungsmodelle werden im Gegensatz dazu systemweit verteilt.

Über die Skalierbarkeit von verteilten Suchmechanismen kann in [70] nachgelesen werden.

Auf die Besonderheit von Inhaltsuche bei Lehrinhalten gehen Eran Segal und seine Co-Autoren in „Learning Module Networks“ [54] ein.

Der im DLE-System vorgeschlagene Suchmechanismus geht davon aus, dass Suchanfragen im DLE sich hauptsächlich auf eingeschränkte Bereiche einer hierarchisch organisierten Datenablage beziehen. Die zugrundegelegte Hierarchie geht davon aus, dass die Inhalte der im System abgelegten Lehrmodule im Sinne einer inhaltsdomänenorientierten ontologischen Struktur eingeordnet sind. Es erscheint hierbei sinnvoll, verschiedene parallel existierende Organisations-Strukturen oder -Mechanismen zu nutzen, um eine möglichst gezielte Suche ausführen zu können. Dies gilt im Besonderen dann, wenn sich die Auswahl der sinnvollsten, das heißt schnellsten oder genauesten Suche, aus dem Kontext der Suchanfrage heraus automatisch treffen lässt.

Da die „angeschlossenen“ Institutionen in der Hauptsache nur in Teilbereichen einer solchen Struktur Inhalte einpflegen oder anfragen, können Suchanfragen gezielt nur an diejenigen Knoten übermittelt werden, welche zu dieser Anfrage auch mögliche Antworten liefern können. Hier ist jedoch die wirkliche Anzahl der zu befragenden Knoten von

der Genauigkeit der Anfrage durch den Nutzer abhängig. Es sollten also auf Grund der Zuordnung und der Profildaten des Nutzers Voreinstellungen für die Suche vorgenommen werden, um eine unnötige Belastung des Systems zu verhindern.

Um die Realisierung beliebiger Strukturierungen und die damit notwendigen unterschiedlichen Suchmechanismen zu ermöglichen und diese leicht im Rahmen des DLE bereitstellen zu können, wird ein generisches Interface für die Suche vorgeschlagen. Auf die verschiedenen Suchdienste kann dann einheitlich zugegriffen werden und je nach Suchanfrage, wie oben beschrieben, der bestmögliche ausgewählt werden. Das entworfene Interface ist in Kapitel 6.2.7 beschrieben.

5.4.5 Caching

Das in System implementierte Caching-Verfahren ist ein statusbehafteter Mechanismus, welcher gleichzeitig eine sichere Abrechnung der zugegriffenen Daten ermöglicht. Die Notwendigkeit eines derartigen Caching-Systems ergab sich hauptsächlich auf Grund der in Kapitel 3 genannten rechtlichen Rahmenbedingungen sowie der Notwendigkeit, ein sicheres Abrechnungssystem zu schaffen.

Der Caching-Mechanismus integriert sich hierbei in das im folgenden Kapitel dargestellte verteilte Filesystem, in welches ein persistenter Caching-Mechanismus eingebettet ist, der den beschriebenen Anforderungen genügt.

5.4.6 Virtuelles File-System (VFS)

Das Virtuelle Filesystem (VFS) des DLE ist ein transparentes, systemweit verteiltes Filesystem, das zur Unterstützung des Austausches aller Arten von Dokumenten und zur Nutzung von Diensten entwickelt wurde. Das VFS ist an den Java Filesystem-Layer und NFSv4 angelehnt. Zusätzlich wird ein Export ausgewählter Teilbereiche in zum Beispiel das Unix-Host-System ermöglicht, um die im vorigen Kapitel dargestellten Anforderungen für die Ablage von anwendungsspezifischen Profildaten zu ermöglichen. Die Anleh-

nung des VFS an NFSv4 bezieht im Besonderen die dort vorgeschlagenen Mechanismen zu Reduktion von Serverplattform-Wartezeiten und Datendurchsatz ein. Die durch Caching und Replikationen von Servern sowie die Funktion der Zugriffskontrolle, über ACL's (Access Control Lists) und Zugriffserfassung zur Nutzungsentgeldberechnung mit ein. Durch die in der Plattform durch den Peer-to-Peer-Charakter vorhandenen Suchmechanismen wird das Caching-Verfahren jedoch wie in dem in Kapitel 2.3.3 beschriebenen „Ocean Store"-Filesystem außerdem noch um den Zugriff auf naheliegende Caches teilnehmender Knoten erweitert. Hierzu muss das in der Arbeit [68] „Entwurf einer AAA-Umgebung für ein auf JXTA basierendes verteiltes Lehrmodell-Datenbanksystem" beschriebene Verfahren zum Sichern und Abrechnen der Cacheinhalte wie folgt erweitert werden:

Der Zugriff auf Cacheinhalte erfolgt im Normalfall wie in Abb. 5.9. Für die oben beschriebene Erweiterung muss für den Zugriff auf die verschlüsselten Inhalte des fremden Caches eine entsprechende Erweiterung implementiert werden. Die Funktionsweise dieser Erweiterung ist in Abb. 5.10 dargestellt.

Die Bilder zeigen den Zugriff einer Anwendung auf ein Datum über dessen, bei der Anwendung bekannten Identifier, seiner UID. Hierbei fragt die Applikation ihren derzeitigen nächsten Server (IKR[1]) an. Das zugeordnete Datum, an der UID zu erkennen, befindet sich an der UT[2]. Im Abb. 5.9 ist der direkte Zugriff auf das Datum an der UT unter Einbeziehung der Caching-Module dargestellt. Abb. 5.10 zeigt die Erweiterung des in [68] beschriebenen Mechanismus um die lokalen Caches. So wird hier beispielhaft der Cache der UKA[3] einbezogen. Es wird hierbei das „Wissen" des erweiterten Cache-Modules genutzt, um den nächstliegenden Replikationsort des gesuchten Datums zu finden, und diesen dann von dort anstelle der UT zu beschaffen. Darüber hinaus zeigen die Bilder noch den Ansatzpunkt für die Abrechnungsmechanismen auf.

Im folgenden Abschnitt soll noch auf Lösungsansätze für allfällige Probleme verteilter Filesysteme eingegangen werden.

[1] Beispielhaft IKR, Institut für Kommunikationsnetze und Rechnersysteme
[2] Bespielhaft UT, University Twente
[3] Beispielhaft UKA: University Karlsruhe

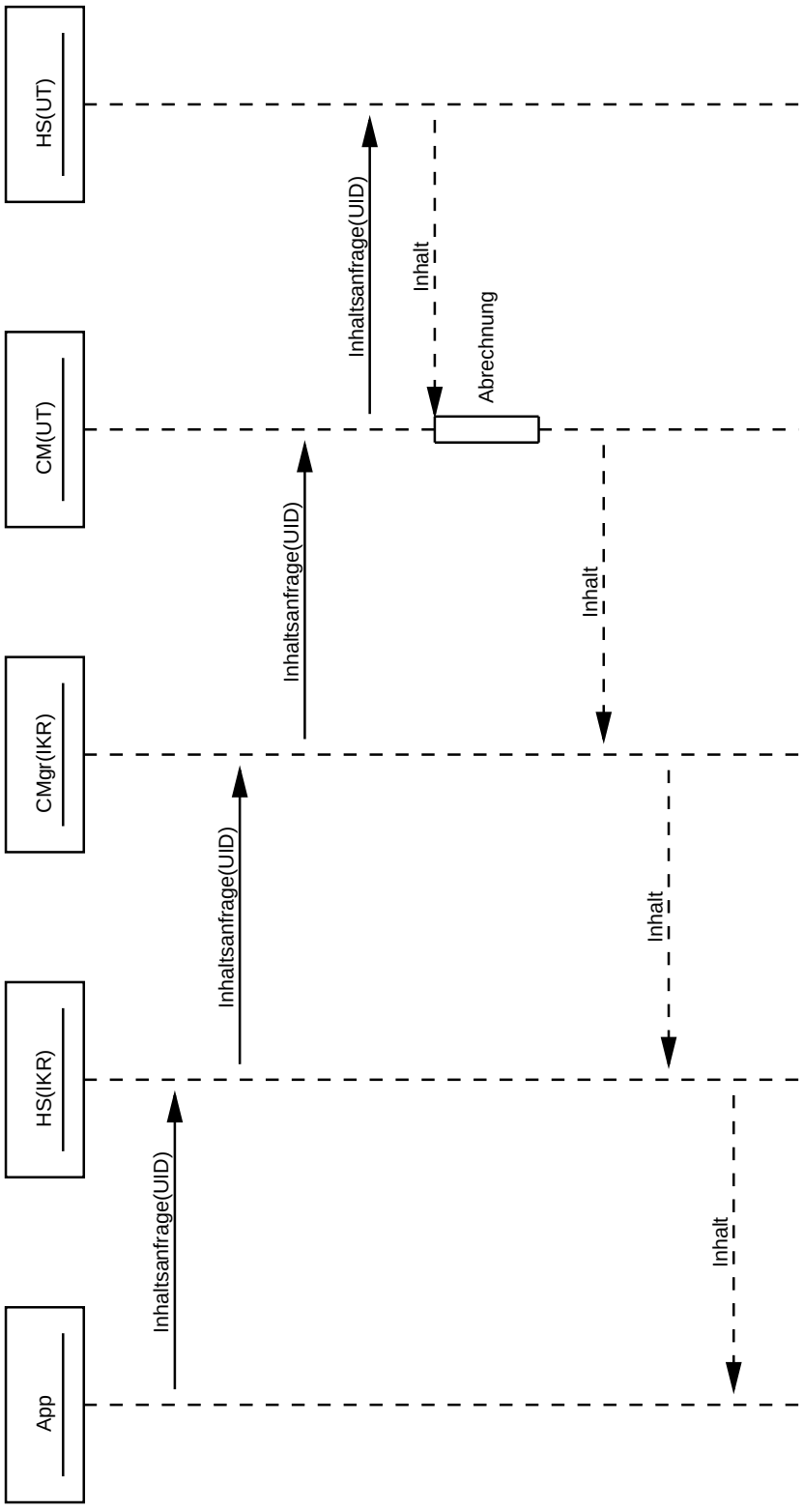


Abbildung 5.9: Zugriff auf lokalen Cache

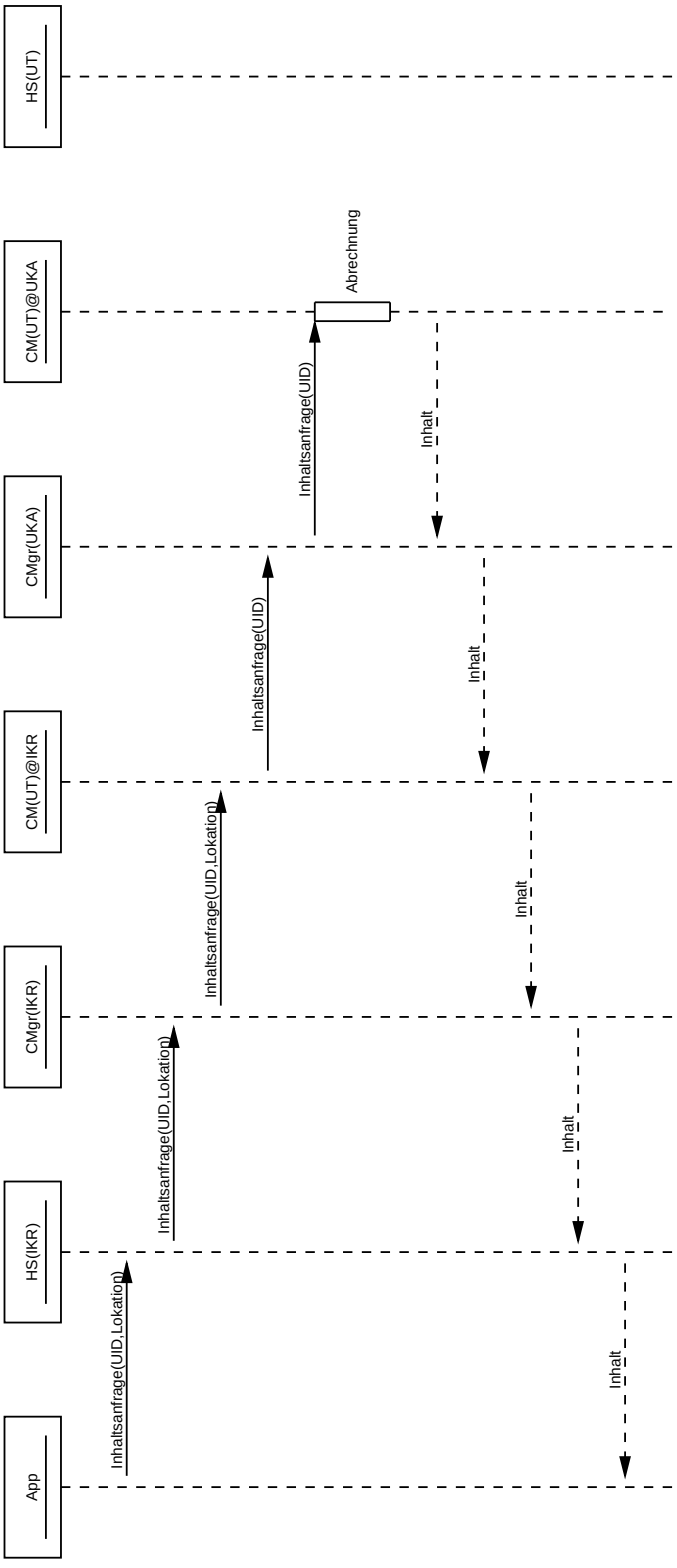


Abbildung 5.10: Zugriff auf verteilte Caches

Ein wesentliches Problem von lose gekoppelten Filesystemen ist die systemweit eindeutige Identifikation der Ressourcen und der Dateien. Der hier vorgeschlagene Mechanismus vergibt systemweit, das heißt über alle Knoten des Serververbundes hinweg, eindeutige File-Identifikatoren, die es ermöglichen, eine Datei systemweit eindeutig durch diesen Identifikator, sei es auf dem originären Server oder in einem entfernten oder lokalen Cache, zu erreichen. Im Falle von NFS werden hierzu für Server und File jeweils ein Datum vom Typ long (64 bit) verwendet, welche nicht notwendigerweise eindeutig sind, und über die Inode Nummern des exportierten Unix Filesystems bestimmt. Im Rahmen des hier vorgeschlagenen VFS werden die mit dem System inhärent eindeutig vergebenen UID's verwendet. Somit kann für das hier dargestellte System dieses Problem als gelöst betrachtet werden.

Die in den folgenden Abschnitten beschriebenen Filesystemtypen der Plattform entsprechen den eben beschriebenen Funktionalitäten.

Filesystem-Typen

Um die Anforderungen der betrachteten E-Learning Applikationen und der Nutzer zu erfüllen sowie die Abbildung möglichst aller Inhaltstypen und Ressourcen zu ermöglichen, werden verschiedene Filesystem-Typen in das System integriert. Diese können statisch oder dynamisch in einen beliebigen lokalen oder entfernten Dateibaum eingeblendet werden. Erreicht werden soll ein alle Systeme übergreifender Mechanismus, welcher flexibel und transparent in der Lage ist, beziehungsweise die Systemnutzer in die Lage versetzt, alle notwendigen Daten über diese Schnittstelle bereitzustellen und auszutauschen.

Lokale Filesysteme

MemFS: Das Memory-Filesystem dient der dynamischen Bereitstellung von Inhalten, Profilen und anderen Daten zum lokalen oder entfernten Zugriff für Server oder Anwendungen. Die im Dateisystem befindlichen Daten werden hierbei nur im Speicher des

Servers gehalten und sollten keine lange Lebensdauer haben. Anwendung findet dieses Filesystem beispielsweise bei der angepassten Bereitstellung von Profil- oder Personalisierungsinformationen.

LocalFs: Das Local-Filesystem dient zum Zugriff und Export persistenter Daten eines beliebigen Teiles des lokalen Filesystems des Servers. Die zusätzlich benötigten Metadaten des Files[4] werden in einer beliebigen SQL-Datenbank abgelegt. Dieses Filesystem eignet sich im Besonderen zur Ablage von schwach strukturierten Materialien oder Applikationsunterstützenden Daten.

SqlDBFS: Das SqlDb-Filesystem dient zur persistenten Speicherung von Daten in einer Datenbank und bietet somit die Möglichkeit des konsistenten, lokalen, verteilten Zugriffes auf diese Daten. Dieses Filesystem ist im besonderen für den flexiblen Zugriff auf stark strukturierte Inhalte wie Lehrmaterialen einsetzbar.

Pseudo-Filesysteme

ObjectFS: Das Object-Filesystem dient zur Bereitstellung beliebiger Inhalte und Funktionalitäten über ein Filesystem. Dieses ist im Sinne von Device-Knoten in Unix zu sehen, jedoch können die Dienste hier dynamisch eingestellt werden. So ist es beispielsweise möglich, einen Socket zu nutzen oder Daten aus oder an beliebigen frei programmierten Quellen zu lesen oder zu schreiben. Dieses ist bei weitem das am flexibelsten einsetzbare Filesystem. Bei einem Netz-Export dieses Filesystems ist jedoch, auf Grund der hohen Datendynamik, kein Caching möglich. Das beschränkt einen sinnvollen Einsatz diese speziellen Filesystems auf den lokalen Bereich oder die Realisierung bestimmter Anforderungen, beispielsweise von virtuellen Laboratorien.

XMLFileFS: Das XMLFile-Filesystem dient zur Abbildung beliebiger XML-Dokumente in ein Filesystem und kann zur Strukturierung von Daten in Directory-Strukturen genutzt werden. Ein Beispiel hierfür ist eine Ontologie-Basisstruktur, die in einem XML-File abgelegt ist. Diese kann dann als Grundlage dynamischer Erweiterungen über das XMLFileFS bereitgestellt werden. Sie dient somit als Basisstruktur für spezialisierte Teile

[4] UID, ACL, usw.

der Ontologie-Struktur, welche die Wissensdomänen der Institutionen oder detailliertere Strukturen abbildet. Ein großer Gewinn ergibt sich hier, wenn dieses Ontologie-Basis-File mit einem hohen Grad verteilt ist, und somit bei allen teilnehmenden Institutionen lokal verfügbar ist. Damit müssen diese häufig verwendeten Daten nicht über das Netz übertragen werden, und die Datenrate im Transportnetz wird reduziert. Der Vorteil wirkt sich hierbei umso stärker aus, je tiefer die Hierarchiestruktur der Ontologie und damit des Filesystembaumes ist, da Suchanfragen damit aufgrund lokal existierender Informationen wesentlich gerichteter gestellt werden können.

OverlayFs: Das Overlay-Filesystem dient der zusammenfassenden Darstellung und „Einblendung“ aller Datentypen in den Dateibaum des jeweiligen Servers. Eine mögliche Struktur eines Dateisystembaumes kann dabei wie in Abb. 5.11 dargestellt aussehen. Die exportierten Dateisysteme entfernter Server werden dynamisch an einer Stelle eingehängt, und dann in die lokale Struktur verlinkt. Die Links werden ebenfalls dynamisch auf Anforderung der entsprechenden Daten angelegt. Greift zum Beispiel ein Nutzer auf ein Lehrmodul zu, so wird dieses entsprechend seiner UID lokalisiert, in das Dateisystem des entfernten Servers eingebunden und dann das entsprechende Modul in die Teilstruktur des Nutzers verlinkt werden. Die in den Links gespeicherten Informationen können zusätzlich noch eine eventuell notwendige Formatumsetzung beschreiben, welche dann beim Zugriff durch den Nutzer transparent die Daten in das von ihm benötigte Format übersetzt.

Entfernte Filesysteme

VFSRemote: Virtual File System Remote Access
Der VFSRemote Mechanismus wickelt den gesamten verteilten Zugriff auf das vom DLE bereitgestellte Filesystem ab. Alle Zugriffe auf die im DLE verteilt vorliegenden Informationen erfolgen für den Nutzer transparent über diese Schicht.

VFS-Nfs-Client: NFS in VFS Import-System
Mit diesem VFS-Dienst ist es möglich, auf Unix-Systemen bereitstehende Daten in den

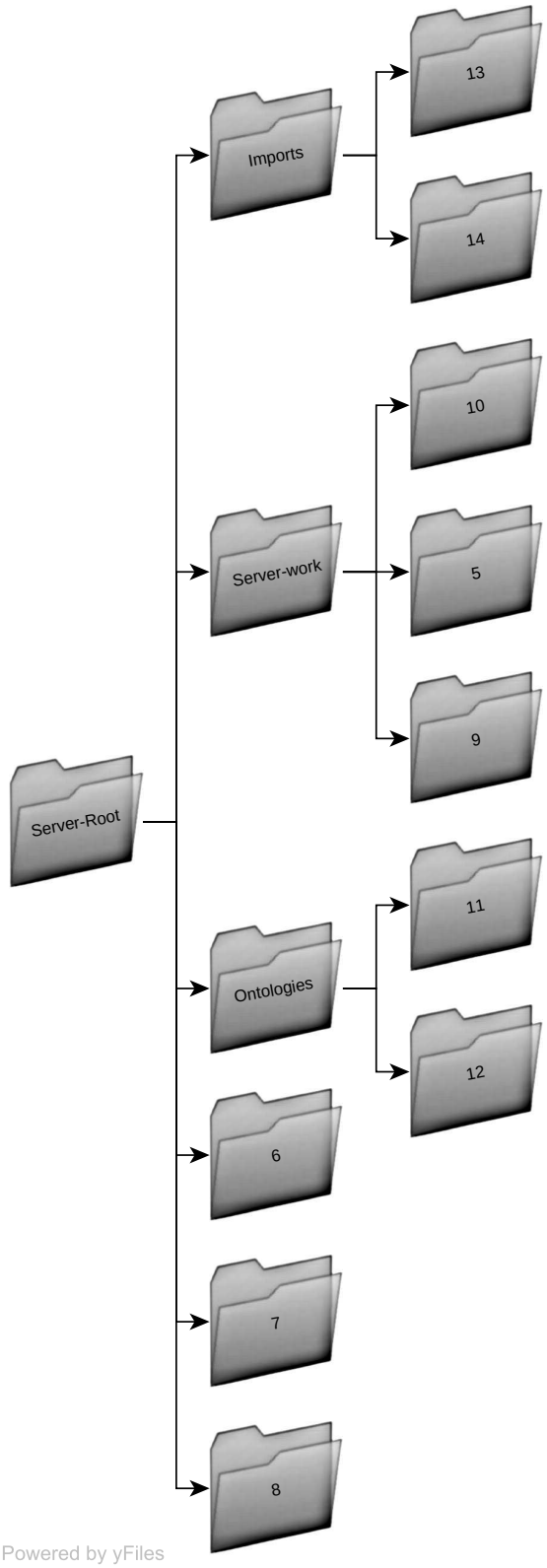


Abbildung 5.11: Overlay-Dateibaumstruktur eines Servers

DLE-VFS-Baum einzubinden. Da die NFS-Identifier jedoch für das DLE-VFS nicht nutzbar sind, wie in Abschnitt 4.6.1 aufgezeigt, wird die Struktur des importierten Host-Filesystems in einer Datenbank parallel aufgebaut und gepflegt. Über diese Indirektion ist eine voll funktionsfähige Anbindung gewährleistet. Wird dieses System an einen NFS-V4 Server angebunden, so ist über die Callback-Methodik ein automatischer Abgleich der Daten zu ihrer Konsistenzhaltung in der Datenbank und im Filesystem möglich. Ist dieses nicht der Fall, so muss durch periodisches Scannen und Vergleichen der beiden Strukturen die Konsistenz erhalten werden. Überlegungen mit diesem System gleichzeitig eine automatisierte Versionierung von Dateien vorzunehmen, werden hier nicht weiter ausgeführt.

Export Mechanismen für DLE-Filesysteme Der Export der Filesysteme kann auf unterschiedliche Arten stattfinden. Die Möglichkeiten unterscheiden sich hierbei hauptsächlich durch die Charakteristik der Anwendung und deren Anforderungen. So ist es möglich, die Filesysteme innerhalb der Plattform durch das VFS-Remote Interface zu exportieren oder außerhalb der Plattform über einen NFS-Übersetzer. Im Folgenden sollen die zwei Möglichkeiten, ihre Unterschiede sowie die Einsatzszenarien betrachtet werden.

NFS-Export: Network File System Export-Funktionalität
Das implementierte NFS-Modul dient zur Bereitstellung von Teilen des DLE-Filesystems im Dateisystembaum eines Unix-basierten Host-Betriebssystem des lokalen Rechners des Anwenders. Diese Möglichkeit ist für die Bereitstellung von Informationen des DLE für Applikationen „außerhalb“ des DLE, welche nur auf diese Weise indirekt auf die für sie notwendigen Informationen zugreifen können. Ein Beispiel für dieses Szenario ist die Bereitstellung applikationsspezifischer Profildaten des Anwenders.

SMB-Export: Windows Filesharing Export-Funktionalität
Da eine reine NFS-Welt nur für Unix-Systeme existiert, sollte auch eine SMB- oder CIFS-Exportmöglichkeit für Windows basierte Systeme bereitgestellt werden. Die Komplexität der Protokolle, sowie das Fehlen verfügbarer Beispiel-Server-Implementierungen las-

sen jedoch eine einfache Implementierung dieser Komponente nicht zu. Für Windows-Systeme kann jedoch auf das im Folgenden dargestellte WebDav-Modul zurückgegriffen werden.

WebDav-Export: WebDav Export-Funktionalität
In Anlehnung an das NFS-Export-Modul wurde zum Einen wegen der eben beschriebenen Betriebssystem-Problematik als auch wegen der interessanten Eigenschaften des WebDav-Protokolls diese Export-Funktionalität implementiert. Diese Eigenschaften wie Versionierung und Annotation, welche durch WebDav spezifiziert werden, bieten interessante Möglichkeiten innerhalb des DLE. Siehe hierzu auch Abschnitt 2.3.3

Sicherheitsmechanismen auf Filesystemebene

Um dem Zugriff auf Dateien oder Dienste, bzw. gesamte Teilbereiche des Filesystems einzuschränken, werden sogenannte Access Control Listen (ACL's) verwendet. Alle Zugriffe auf Ressourcen werden mittels der ACL kontrolliert und über die Caching-Schicht des VFS abgewickelt, welche Teile der Zugriffskontrolle sowie die Abrechnung der in Anspruch genommenen Ressourcen abwickelt.

5.5 Die Datenablagestruktur des DLE

Aus den in den letzten Kapiteln beschriebenen Sachverhalten ergibt sich die im folgenden kurz dargestellte interne Datenstruktur des vorgeschlagenen Systems. In Abb. D.3 sind alle Zusammenhänge ersichtlich. Die Abbildungen 5.12 und 5.13 zeigen Ausschnitte aus der Strukturdefinition zur Erstellung der notwendigen Datenbank-Tabellen, Programm-Klassen und Graphen.

```
Attr(login,DTVC(10))
Attr(username,DTVC(30))
Attr(memoid,DTVC(10))
Attr(homeinst,DTVC(20))
Attr(NAME,DTVC)
Attr(TYP,DTVC)

AttrList(learningObject,NAMEAttr,TYPAttr)
AttrList(SYSTEM,NAMEAttr,TYPAttr)
AttrList(user,loginAttr,usernameAttr,memoidAttr,homeinstAttr)

Attr(stylesheet,DTVC(100))
Attr(transformEngine,DTVC(100))
Attr(srcMime,DTVC(20))
Attr(destMime,DTVC(20))
AttrList(LayoutHint,stylesheetAttr,transformEngineAttr,srcMimeAttr,destMimeAttr)

CreDBO(LearningObject,DBObj,learningObjectAttList)

CreDBOV(Course,LearningObject)
CreDBOV(Module,LearningObject)
CreDBOV(Atom,LearningObject,learningObjectAttList)
CreDBOV(Glue,LearningObject)

CreDBO(LayoutHint,DBObj,LayoutHintAttList)

CreDBO(SYSTEM,DBObj,SYSTEMAttList)

CreDBO(LOMASTER,DBObj)
CreDBO(METADATA,DBObj)
CreDBO(MDMASTER,DBObj)
CreDBO(NOTE,DBObj)
CreDBO(Description,DBObj)
CreDBO(CACHE,DBObj)
CreDBO(JAVAOBJECT,DBObj)
CreDBOV(CACHEMANAGER,JAVAOBJECT)
CreDBOV(CACHEMODUL,JAVAOBJECT)
CreDBOV(LOGINMANAGER,JAVAOBJECT)
CreDBOV(LOGINMODUL,JAVAOBJECT)
CreDBOV(FILESYSTEM,JAVAOBJECT)
CreDBO(StorageLocation,DBObj)
CreDBO(ACL,DBObj)
CreDBO(Data,DBObj)
```

Abbildung 5.12: Interne Datenstrukturdefinition des DLE-Systems

```
CreDBOV(DataObj,Data)
CreDBOV(DataStorageObj,Data)
CreDBO(User,DBObj,userAttList)
CreDBOV(Author,User)
CreDBOV(Teacher,User)
CreDBOV(Learner,User)

CreDBO(Profile,DBObj)
CreDBO(Preferences,DBObj)
CreDBO(PreKnowledge,DBObj)

CreDBO(ApplicationData,DBObj)
CreDBO(Institution,DBObj)

CreDBRel(SEQUENCE,DBRel,LearningObject,LearningObject)
CreDBRel(START,DBRel,LearningObject,LearningObject)
CreDBRel(FIN,DBRel,LearningObject,LearningObject)
CreDBRel(BRANCH,DBRel,LearningObject,LearningObject)
CreDBRel(LOTOLO,DBRel,LearningObject,LearningObject)
CreDBRel(LOUSES,DBRel,LearningObject,LOMASTER)
CreDBRel(LOTOLOMSTR,DBRel,LearningObject,LOMASTER)
CreDBRel(LOMSTRTOLOMSTR,DBRel,LOMASTER,LOMASTER)
CreDBRel(LOMSTRTOLOOBJ,DBRel,LOMASTER,LearningObject)
CreDBRel(LOMSTRTOMDMSTR,DBRel,LOMASTER,MDMASTER)
CreDBRel(LOTOMD,DBRel,LearningObject,METADATA)
CreDBRel3(LoToDescr,DBRel,LearningObject,Description,User)

CreDBRel(MDMSTRTOMDOBJ,DBRel,MDMASTER,METADATA)
CreDBRel(LModToLMgr,DBRel,LOGINMODUL,LOGINMANAGER)
CreDBRel(CModToCMgr,DBRel,CACHEMODUL,CACHEMANAGER)
CreDBRel(SysToFS,DBRel,SYSTEM,FILESYSTEM)
CreDBRel(SysToCache,DBRel,SYSTEM,CACHE)
CreDBRel(SysToLMgr,DBRel,SYSTEM,LOGINMANAGER)
CreDBRel(SysToCMgr,DBRel,SYSTEM,CACHEMANAGER)
```

Abbildung 5.13: Interne Datenstrukturdefinition des DLE-Systems

```
CreDBRel(DoToLo,DBRel,Data,LearningObject)
CreDBRel(DoToACL,DBRel,Data,ACL)
CreDBRel(DoToFs,DBRel,Data,FILESYSTEM)
CreDBRel(CacheToFs,DBRel,CACHE,FILESYSTEM)
CreDBRel(CacheToStoreLoc,DBRel,CACHE,StorageLocation)
CreDBRel(FsToStoreLoc,DBRel,FILESYSTEM,StorageLocation)
CreDBRel3(LoToNote,DBRel,LearningObject,NOTE,User)
CreDBRel(LoToUser,DBRel,LearningObject,User)

CreDBRel(AtomToLayout,DBRel,AtomClass,LayoutHint)

CreDBRel(UserToPref,DBRel,User,Preferences)
CreDBRel(UserToInst,DBRel,User,Institution)
CreDBRel(UserToPreK,DBRel,User,PreKnowledge)
CreDBRel(UserToAppD,DBRel,User,ApplicationData)
CreDBRel(UserToProf,DBRel,User,Profile)
```

Abbildung 5.14: Interne Datenstrukturdefinition des DLE-Systems

Kapitel 6

Simulation und prototypische Realisierung zur Abschätzung der Systemeigenschaften

Um Untersuchungen zum Verhalten der vorgeschlagenen Systemarchitektur und seiner Kommunikationsanforderungen durchführen zu können, wurden die wesentlichen Teile des Systems simuliert oder prototypisch realisiert. Die Teilbereiche und Spezifika dieser Implementierungen werden im Folgenden beschrieben. Hierbei wird zuerst auf die realisierten beziehungsweise simulierten Teilbereiche eingegangen, um diese später in den Gesamtkontext einzuordnen und die Zusammenhänge, unter Einbezug verschiedener Abschätzungen, zur Bewertung darzustellen. Die Subsysteme werden kurz beschrieben und die Begründung für die jeweilige Implementierung angegeben. Darauf folgend wird das Zusammenspiel der einzelnen Subsysteme beschrieben und auf deren Abhängigkeiten eingegangen. Zuletzt werden die verschiedenen Teilsysteme dargestellt und in das Gesamtsystem eingeordnet und nachfolgend deren Relevanz und die sich ergebenden Bewertungsmöglichkeiten aufgezeigt.

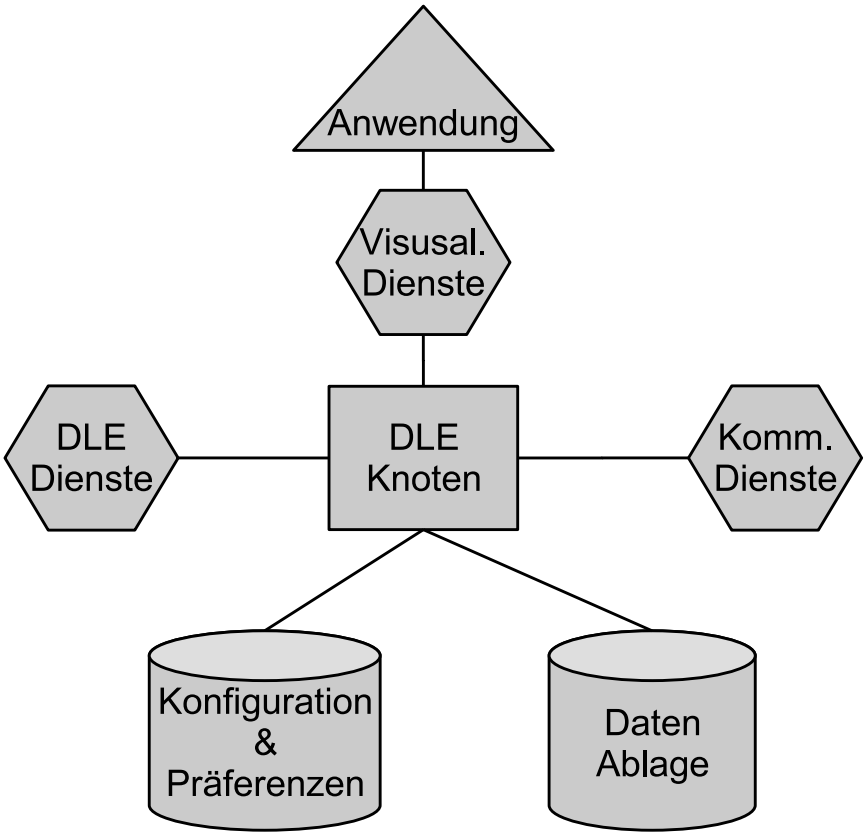


Abbildung 6.1: Grundstruktur des Systems

6.1 Modellierung des Systems und seiner Teileigenschaften

Zur Modellierung des Systems zur Abschätzung der benötigten Netzbandbreite wurde das System analysiert und in seine wesentlichen zum Kommunikationsaufwand beitragenden Teilsysteme zerlegt.

Um die Analyse der Datenflüsse sinnvoll durchführen zu können, wurden jene Teilsysteme der vorgeschlagenen Architektur so ausgewählt, dass am Ende jeweils die Ergebnisse eines oder mehrerer Teilsysteme in einem definierten Zusammenhang untersucht werden können. Für einen Überblick über die ausgewählten Systeme siehe Abb. 6.1. Zudem wurden nur diejenigen Teilsysteme untersucht, welche in der Hauptsache, auf Grund ihrer Kommunikationscharakteristika, den Datenverkehr des DLE bestimmen. Dazu wurden die nachfolgend aufgelisteten Teilsysteme untersucht.

6.1.1 Modellierung der SRDI DHT-Implementierung

Die in JXTA eingesetzte DHT-Variante namens SRDI[1] wird durch eine Implementierung des Mechanismus mittels Simulation untersucht. Hierbei zeigt sich, dass die Netzbandbreite im Durchschnitt eine unproblematische Größe darstellt, da die benötigte Bandbreite für lange Zeiten gering ist. Ein Problem sind jedoch zeitlich auftretende Spitzen im Verkehrsaufkommen, welche das System stark behindern. Um diese Problematik der instationären Simulation zu lösen, wurden in der vorliegenden Arbeit die Warteschlangenfüllstände aller Knoten über die Zeit aufgezeichnet und ausgewertet. Hierzu wurde die Simulationsumgebung um ein Datenbankinterface erweitert, um die anfallenden Daten strukturiert und performant aufzeichnen und auswerten zu können.

6.1.2 Übermittlung von Login-Daten, Pseudonym-Daten und Login-Modulen

Das DLE-System nutzt für einen Anmeldevorgang für jeden Nutzer ein so genanntes Login-Modul (Abschnitt 5.4.2), welches über die in Abschnitt 5.2 beschriebenen Protokolle kommuniziert. Die Größe des zu übertragenden Login-Moduls im Prototyp ist ca. 100 KByte. Es sollte hierbei jedoch nicht nötig sein, für jeden Nutzer ein „neues“ Modul von dessen Heimatserver anzufordern. Die für die Kommunikation notwendige Datenmenge für die Authentisierung und Autorisierung des Nutzers beläuft sich im Prototyp auf durchschnittlich 300 KByte und bestimmt sich direkt aus den in Abschnitt 6.4.4 dargestellten Protokollabläufen und Größen der zu übertragenden Softwaremodule. Die durchschnittlich pro Login- bzw. Pseudonym-Update-Vorgang zu übertragende Datenmenge beläuft sich im Prototyp auf die in Tabelle 6.1 aufgelisteten Werte.

[1] Shared Resources Distributed Index(SRDI) ist die in JXTA eingesetzte DHT - Variante

Tabelle 6.1: Größenabschätzung der zu übermittelnden Daten

Aktion	Datenmenge
Login	300 KByte
Pseudonym-Update	50 KByte

6.1.3 Übertragung von Personalisierungsdaten

Die Übertragung von Personalisierungsdaten findet über das in Abschnitt 5.4.6 vorgestellte verteilte Filesystem statt. Da die Anzahl der notwendigen Personalisierungsdaten nicht von den im DLE-System festgelegten Strukturen und Datentypen bestimmt oder eingeschränkt, sondern maßgeblich durch die verwendete Anwendung bestimmt wird, kann hier keine exakte Abschätzung vorgenommen werden. Geht man aber davon aus, dass sich die genutzten Anwendungen, unterschieden nach Nutzergruppen, verhalten wie die in Anhang C aufgelisteten Beispielanwendungen, welche ein breites Spektrum von Autorensystemen sowie personalisierbaren Lernportalen und unterstützenden Anwendungen abdecken, so belaufen sich die durchschnittlich zu übertragenden Datenmengen auf 30-1024 KByte.

6.1.4 Übermittlung von Caching-Daten und -Modulen

Caching Module werden in der Regel nur einmal pro „Lebensdauer“ eines Servers auf Anforderung übertragen. Die Hauptlast aus diesem Systemteil ergibt sich nur durch die Updates und den Abgleich der Schlüssel des Caching-Moduls mit seinem Heimatserver und nicht durch die eigentliche Modulübertragung. Um bei häufigen Neustarts von Servern durch die Modulübertragung keine unnötigen Datenübertragungen zu verursachen, kann hier noch ein Mechanismus zur Wiederanmeldung bereits erhaltener Cache-Module realisiert werden. Hierbei tritt jedoch das Problem der Remote-Code-Verifikation auf, welches durch die in Abschnitt 5.4.2 beschriebene Erweiterung gelöst wurde. Die Basis der gewählten Lösung findet sich in der Implementierung in [68].

6.1.5 Übermittlung von Accounting-Daten

Die Übertragung der gesammelten Accountingdaten kann zu einem beliebigen Zeitpunkt stattfinden. Wählt man den Zeitpunkt so, dass die Übertragung zu Zeiten stattfindet, an dem das System nur mit wenig Verkehr durch Nutzerinteraktion belastet ist, z.B. nachts, so können diese Bandbreitenanteile aus der hier dargelegten Betrachtung herausgelassen werden.

Die Größe der zu übertragenden Datenmenge hängt hier maßgeblich von den folgenden Faktoren ab:

- Verwendung fremder Materialien
- Hohe und verteilte Verwendung eigener Materialien
- Verteilung der Nutzer auf fremde Server

6.1.6 Datenaufwand des Virtuellen Filesystems

Aus den in den Einzelbetrachtungen gewonnen Ergebnissen kann nun der Protokolloverhead des VFS-Protokolls und die Reduktion der zu übertragenden Datenmenge durch die Cache-Mechanismen einbezogen werden. Wird das System hinreichend stabil betrieben, ergibt sich auf Grund des inhärenten persistenten Caching und der Annahme, dass die Zugriffe nicht stark über die Inhalte streuen, dass Lehrmodule nur sehr selten übertragen werden müssen. Gleiches gilt für die Präferenz- und Profildaten, so der Nutzer keine häufigen Veränderungen an ihnen vornimmt.

6.2 Einordnung der betrachteten Teilsysteme

In den folgenden Abschnitten werden Ansätze zur Annäherung an eine Architektur-Bewertung erarbeitet und aufgezeigt. Die Komplexität des Gesamtsystems und seiner Teilsysteme stellt eine hohe Schwelle für eine belastbare Bewertung dar und kann auch

in der vorliegenden Arbeit nicht erreicht werden. Vielmehr werden Möglichkeiten aufgezeigt, welche die Bewertung von Teilsystemen zulassen und Anstöße für weitergehende Forschungen sind. Die hier betrachteten Teilbereiche umfassen eine Abschätzung der Größe der Lehrmoduldaten anhand der im CANDLE-Projekt erstellten Module, der Anzahl der Systemteilnehmer (Institutionen wie Anwender), der duch das Peer-to-Peer-System zu verwaltenden Anzahl an Datensätzen, die Simulation des Verhaltens des DHT-Mechanismus des JXTA Peer-to-Peer-Systems sowie der Aufbau eines Messsystems für beliebige, real existierende Systeme.

Die untersuchten Teilsysteme bilden die minimal notwendige Menge an Modulen, um die Funktionalität und den Kommunikationsaufwand des Gesamtsystems abschätzen zu können. Es wurden die Basiskommunikationsmechanismen, der Filesystem-Layer sowie die Basisfunktionalitäten der Sicherheitsinfrastruktur realisiert, um damit die Funktionalität nachzuweisen. Alle weiteren Dienste und Applikationen setzen auf diese Mechanismen auf oder nutzen sie, und nur spezielle, jedoch abschätzbare, anwendungsspezifische Kommunikationsstrukturen sind darüberhinaus notwendig, um ein voll funktionsfähiges Gesamtsystem zu erhalten. Damit steht eine ausreichende Grundlage zur Bewertung der Kommunikationseigenschaften zur Verfügung. In den folgenden Abschnitten werden nun die einzelnen Teilsysteme und Dienste in den Kontext des Gesamtsystems eingeordnet und die Aussage der damit erreichten zur Bewertung notwendigen Funktionalität belegt.

6.2.1 Bewertungskriterien für das Server-System

Die sich letztendlich stellende Frage ist diejenige nach sinnvollen, belastbaren Kriterien, an denen sich das vorgeschlagene System messen beziehungsweise bewerten lässt. Die Kriterien ergeben sich aus der Betrachtung der folgenden Punkte.

- P2P Management Overhead
  Hier wäre die Frage zu beantworten, ob das als Transportmechanismus gewählte Overlay-Netz prinzipiell die Skalierbarkeitsanforderungen erfüllen kann, oder ob es

eigentlich von vornherein ungeeignet ist. Gibt es eine andere Möglichkeit die „problemlose“ Kommunikation zwischen den Systemen herzustellen? Im Anschluss stellt sich weiterhin die Frage, ob das P2P-System so gestaltet werden kann, dass es die bereits beschriebenen Anforderungen erfüllen kann. Im Besonderen besteht das Problem, dass auch das P2P-System nicht ohne „offene“ Dienstanbieter auskommt. Diese „offenen“ Dienstanbieter sind am DLE-Verbund teilnehmende Institutionen, die auf Grund ihrer Netzinfrastruktur direkt, das heißt ohne von einer Firewall abgeschirmt zu sein, die DLE- und Peer-to-Peer-Dienste bereitstellen. Über diese Server, so sie diese Aufgabe wahrnehmen, läuft dann die gesamte Kommunikation der „nicht offenen“ Server. Hier muss dann die Frage beantwortet werden, wie groß die an diesen Instanzen anfallende Last wäre. Diese Diensteanbieter sind notwendig, um Peers, welche hinter Firewalls und Adress-Übersetzern „verborgen“ sind, in der Form von Rendezvous-Peers und/oder Relays eine Kommunikation mit dem System zu ermöglichen. Deren Funktionalität kann in [23] nachgelesen werden.

- Dokumentengrößen und Dokumentenverteilung

  Die sich hier stellende Frage ist, mit welchen Datenmengen und -größen ein solches System umgehen muss, und in wie weit diese die Menge der zu übermittelnden Informationen, und somit Verteilnetz-Last, beeinflusst. Hier kann evaluiert werden, ob der vorgeschlagene verteilte Cachingmechanismus dazu geeignet ist, die Last im Mittel wesentlich zu reduzieren. Entsprechende Überlegungen und Untersuchungen wurden auch im Rahmen des Ocean Store-Projektes angestellt und sind in [3] nachzulesen.

- Managementoverhead des Gesamtsystems

  Die vorgeschlagene Architektur ist bei der angenommenen Auslegung durch die Hierarchisierung der Gruppenstruktur und der damit erreichten „Lokalisierung“ der Teilnehmerknoten in der Lage, die gestellten Anforderungen zu erfüllen. Aus dieser Hierarchisierung ergibt sich, dass das System in einen nahezu statischen „zentralen“ Bereich und einen davon getrennten, aber jeweils kleinen, lokalen Bereich mit

höherer Knotenfluktuation zerfällt. Damit bleiben die erhöhten Managementlasten auf lokale Bereiche beschränkt und haben keinen Einfluss auf die Weitverkehrsebene des Systems.

- Einfluss des A4C-Subsystems
  Das A4C-Subsystem zerfällt in die wesentlichen Teile des Logins, der Pseudonymverwaltung, der Abrechnungsmechanismen sowie der Zugriffskontrollmechanismen, deren Datenkommunikationsaufwände bei großer Nutzerzahl durchaus beachtet werden müssen. Gerade bei den Abrechnungsmechanismen, in welche sowohl Post- als auch Prepaid-Verfahren einbezogen werden müssen, um die geforderten Eigenschaften des Sytems zu erfüllen, können große Datenmengen entstehen, welche wesentlich das Datenaufkommen beeinflussen.

- Einfluss der Caching-Mechanis
  Der Zugriff auf verteilte Ressourcen und Inhalte bedeutet auch deren Transport über das Kommunikationsmedium. Diese Datenmenge zu reduzieren ist die Aufgabe der Cachingmechanismen, deren geeignete Wahl und deren „Treffsicherheit“ sind ein Hauptkriterium bei der Betrachtung des Datenaufkommens des Gesamtsystems.

Diese Kriterien sollen nun auf die zu bewertenden Subsysteme angewandt und, die sich daraus ergebenden Aussagen, kurz dargestellt werden.

6.2.2 Peer-to-Peer-Nachrichtenaustausch

Die Übertragung von Daten und Informationen innerhalb des DLE findet über ein unidirektionales Nachrichtenaustauschsystem statt. Alle weiteren Kommunikationsmechanismen werden innerhalb des DLE-Systems auf dieser Basis abgebildet. Die Grundmechanismen des Peer-to-Peer-Systems bedienen sich hierbei ebenfalls des unidirektionalen Nachrichten-Austauschsystems. Somit können alle Betrachtungen bezüglich des Kommunikationsaufkommens auf diese Ebene abgebildet werden und alle darunterliegenden Protokolle und Transportmechanismen abstrahiert werden.

6.2.3 Der Systemkern (Micro-Kernel)

Für den in Abschnitt 5.1 entworfenen Systemkern (MikroKernel), welcher die Dienste kontrolliert, wurden die wichtigsten Basisfunktionalitäten realisiert. Diese sind im einzelnen:

- Die Anbindung an das Peer-to-Peer-System
- Die Kommunikations-Schicht
- Die Filesystem-Schichten
- Die Nfs-Export-Funktionalität
- Die Datenbankanbindung
- Ein Suchmechanismus auf Basis des DLE-Datenmodells

6.2.4 Das AAA-Teilsystem

Die dem AAA- und Abrechnungs-System zu Grunde liegenden Basismechanismen wurden in [68] entwickelt und beschrieben. Hierbei wurden die Mechanismen und die Kommunikationsmechanismen an die Randbedingungen des Serversystems angepasst und die dort beschriebenen Mechanismen in den Prototypen integriert. Die durchgeführten Erweiterungen betreffen vor allem die Anpassung an die erweiterte Caching-Funktionalität des DLE (siehe Abschnitt 5.4.5). Die Integration erforderte darüberhinaus auch die im Rahmen der erwähnten Arbeit nicht vollständig ausgeführte Entwicklung der Remote-Code-Verification, da dieses für die Abschätzung des Kommunikationsaufwandes und zum Beweis der prinzipiellen Machbarkeit notwendig wurde.

6.2.5 Realisierung der Transportschicht

Für die Transportschicht wurde ein an UDP angelehntes Verfahren implementiert, das sich in die Struktur des DLE eingliedert, und für die Applikation transparent ist, dass

heißt, eine Abstraktion des Nachrichtenaustausches der Peer-to-Peer-Plattform bildet. Damit bietet sie der DLE-Plattform und den angeschlossenen Anwendungen die Möglichkeit, völlig unabhängig von der „physikalischen“ Kommunikationsschicht, Dienste bereitzustellen oder zu nutzen.

Auf die Realisierung des TCP-Transport-Mechanismus wurde aufgrund seiner Komplexität verzichtet, da sich die notwendigen Aussagen über dessen Kommunikationsverhalten auch aus dem UDP-Modell ableiten lassen. Dies gilt, da der DLE TCP-Transport-Mechanismus ohne einen Flusskontrollmechanismus vorgesehen ist. Siehe auch Abschnitt 5.3.1

6.2.6 Realisierung der Filesysteme

Die Realisierung des VFS orientiert sich an den Mechanismen der in Abschnitt 2.3.3 beschriebenen verteilten Filesysteme. Die in der vorliegenden Arbeit verfolgten Ziele setzen eine Mischung aus Eigenschaften und Mechanismen aller vorgestellten Filesysteme voraus. Die prototypische Realisierung bleibt aus Aufwandsgründen hierbei nahe an NFS-V3, bezieht jedoch die Mechanismen für Access Control Lists, Cache Management und Token-basierten Sicherheitsmechanismen mit in die Betrachtungen ein. Eine Beschreibung dieser Punkte findet sich in den Kapiteln über das DLE-Caching sowie über die Authentifizierungs- und Sicherheitsmechanismen.

Die Implementierung erhebt nur den Anspruch eines Proof-of-Concept, im besonderen für das Overlay-Filesystem, welches es den Nutzern des DLE ermöglicht, die Inhalte aller angeschlossenen Institutionen in einem virtuellen Dateibaum zu betrachten.

Die durch diese spezielle Implementierung benötigte Datenverkehrsmenge dient im Rahmen der Arbeit zu einer groben Abschätzung der im Worst-Case benötigten Bandbreite für den Zugriff auf Inhalte über das VFS, aus welchem dann die durchschnittlich benötigte Bandbreite im Betrieb abgeschätzt werden kann.

6.2.7 Realisierung des Such-Dienstes

Auf eine spezielle Realisierung des Suchdienstes wurde im Rahmen der Arbeit verzichtet, da dieser Themenbereich zu viele Facetten aufweist, um ihn hier geschlossen betrachten zu können.

Inhärent stellt jedoch die für den Prototypen gewählte Infrastruktur JXTA einen Suchdienst zur Verfügung [24], welcher die effektive Suche in den dem DLE zur Verfügung stehenden Metadaten ermöglicht. Dieses wird dann besonders effektiv, wenn dieser Suchmechanismus durch die in Abschnitt 5.4.4 vorgeschlagenen Strukturierungsmöglichkeiten ergänzt wird. Durch die Kombination von Datenstrukturierung und Peer-to-Peer Suchmechanismen sowie die Strukturierung des DLE in „abgeschlossene“ Bereiche wird erreicht, das sowohl eine Suchanfrage gezielt abgesetzt und bearbeitet wird, als auch die Antworten zeit- und themennah dem Anwender zur Verfügung stehen.

6.3 Durchführung der Simulation und Messung

Die Komplexität der Aufgabe, das DLE-System anhand der aufgezeigten Kriterien zu bewerten, lässt keine in sich geschlossene Evaluation des Gesamtsystems zu. Um dennoch an die notwendigen Daten für eine Bewertung zu gelangen, wurden Teile des Systems simuliert und ergänzend dazu Messungen an dem erstellten Prototypen durchgeführt. Die folgenden Teilkapitel geben einen Einblick in diese Aktivitäten.

6.3.1 Simulation des Peer-to-Peer Netzes

Die ereignisbasierte Simulationsumgebung, welche für die Bandbreite-Bestimmung des DHT-Mechanismus genutzt wurde, basiert auf dem Port-Konzept der IKR eigenen Simulationsumgebung (SimLib), ist jedoch eine im Rahmen der vorliegenden Arbeit entstandene, völlige Neuimplementierung dieses Konzeptes in Java. Hierbei wurden auch weitergehende Konzepte, wie die dynamische Kontrolle des Simulationsvorganges zur

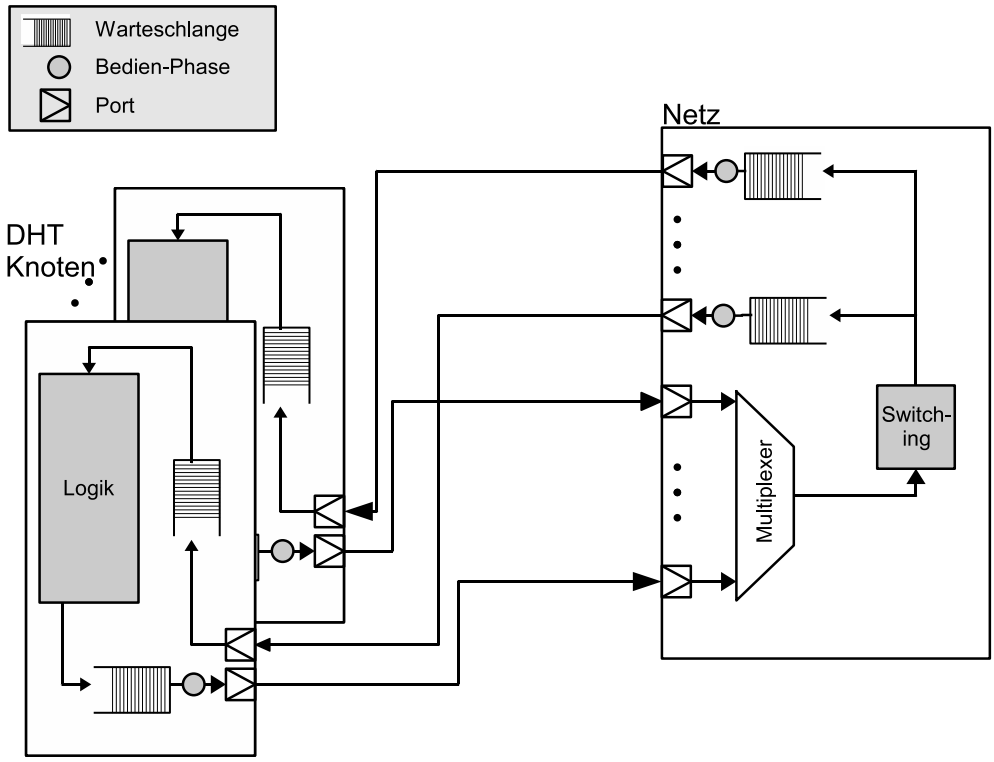


Abbildung 6.2: Simulationsmodell

Laufzeit, freie Konfiguration des Simulationsmodells zur Laufzeit sowie ein Bussystem zur Kommunikation der Simulationskomponenten integriert. Durch die konsequente Implementierung des „Inversion of Control“-Paradigmas entstand so parallel zu den Forschungen eine hochflexible Simulationsumgebung, die die Möglichkeit bietet, die Java-Module des DLE-Systems direkt simulativ zu bewerten, bzw. deren Verkehrseigenschaften zu bestimmen.

Simulationsmodell

Zur Bestimmung der benötigten Bandbreite des DHT-Mechanismus wurde eine Emulation des SRDI-Mechanismus auf der Basis einer ereignisgesteuerten Simulationsumgebung [Anhang A] implementiert. Bild 6.2 zeigt das Modell der Knoten und des sie verbindenden Netzes.

Das System wird mit verlustfreien Warteschlangen simuliert. Werden im DLE-System keine Relays genutzt, so darf diese Annahme getroffen werden, da die in der unterlie-

genden Transportschicht genutzten Protokolle (TCP) Verluste auf der hier betrachteten Ebene verhindern.

Sollten Relays im System eingesetzt werden, um Firewalls oder NAT[2]-Grenzen zu überwinden, so lassen sich die entstehenden Verluste in dessen Warteschlangen leicht durch Anpassung der Warteschlangen im Simulationsmodell nachbilden.

Zudem kommt beim Einsatz von Relays ein zusätzlicher Overhead zum Kommunikationsaufwand des Peer-To-Peer-Basissystems, und damit des DLE-Systems, hinzu. Für diejenigen Peer-Knoten, welche aufgrund ihrer Netzanbindungs-Randbedingungen, nur über ein Relay zu erreichen sind, müssen zusätzliche Informationen über dessen Erreichbarkeit im System verteilt werden. Dies geschieht durch sogenannte Routing-Advertisements, welche in Analogie der sonstigen Advertisements verteilt werden. Durch diese zusätzlichen Managementdaten im System steigt auch dessen, durch Managementdaten verursachter, Kommunikationsaufwand.

Ferner müssen die Eigenschaften der DLE-Serversysteme, das heißt ihre Gruppenstruktur, in die Betrachtung mit einbezogen werden. Jede Gruppe, in welcher ein Serversystem eingebucht ist, verhält sich wie einer der in Abb. 6.2 dargestellten DHT-Knoten. In der Bewertung des durchschnittlichen Verkehrsaufkommens werden, um diese Eigenschaft zu modellieren, Knoten entsprechend der Gruppenanzahl eingebuchter Gruppen überlagert.

Das Simulationsmodell geht von einem zentralen Transportnetzkonten („Netz“) aus, an welchem die Peer-to-Peer-Knoten („DHT-Knoten“) angeschlossen sind. Die Switching-Komponente des Netz-Knotens übernimmt hierbei die Verteilung der Nachrichtenpakete an den jeweils angesprochenen Knoten. Die Logikkomponente dagegen übernimmt die Verarbeitung der eingehenden Pakete und implementiert die Eigenschaften der „Distributed Hashtable“, das heißt im vorliegenden Fall, die Verarbeitung nach dem Algorithmus der in JXTA eingesetzten SRDI-Variante einer DHT. Die in der vorliegenden Arbeit verwendete Implementierung ist in den Abbildungen 6.3, 6.4, 6.5, 6.6, 6.7, und 6.8 dargestellt die während der Simulation eingesetzte Parametrisierung ist in Abb. 6.9

[2]Network Address Translation

wiedergegeben.

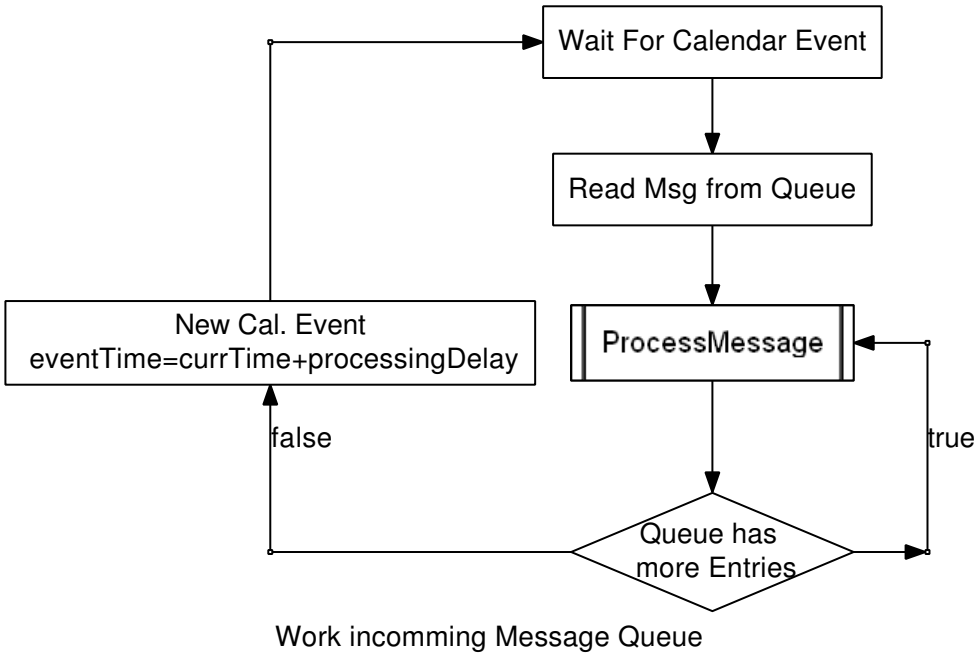


Abbildung 6.3: Die Logik der SRDI-Implementierung (Task zum bearbeiten eingehender Packete)

Simulationsumgebung und Durchführung

Die einzelnen Simulationsschritte wurden auf den Rechnern des Instituts mit der Parametrisierung der in Abb. 6.10 auszugsweise angegebenen Steuerdatei und der in Abb. 6.9 angegebenen DHT-Knoten-Parametrisierung berechnet. Aufgrund des hohen Hauptspeicherbedarfes der Simulation bei hohen Knotenzahlen und großen Datensatzmengen musste zusätzlich eine Datenbank für die Speicherung der Routing- und DHT-Tabellen angebunden werden. Die Simulationszeiten für einige Einzelpunkte beliefen sich dadurch auf Zeiten größer als eine Woche. Dadurch waren weitergehende Untersuchungen als die in Abschnitt 6.4 im Rahmen der vorliegenden Arbeit nicht möglich.

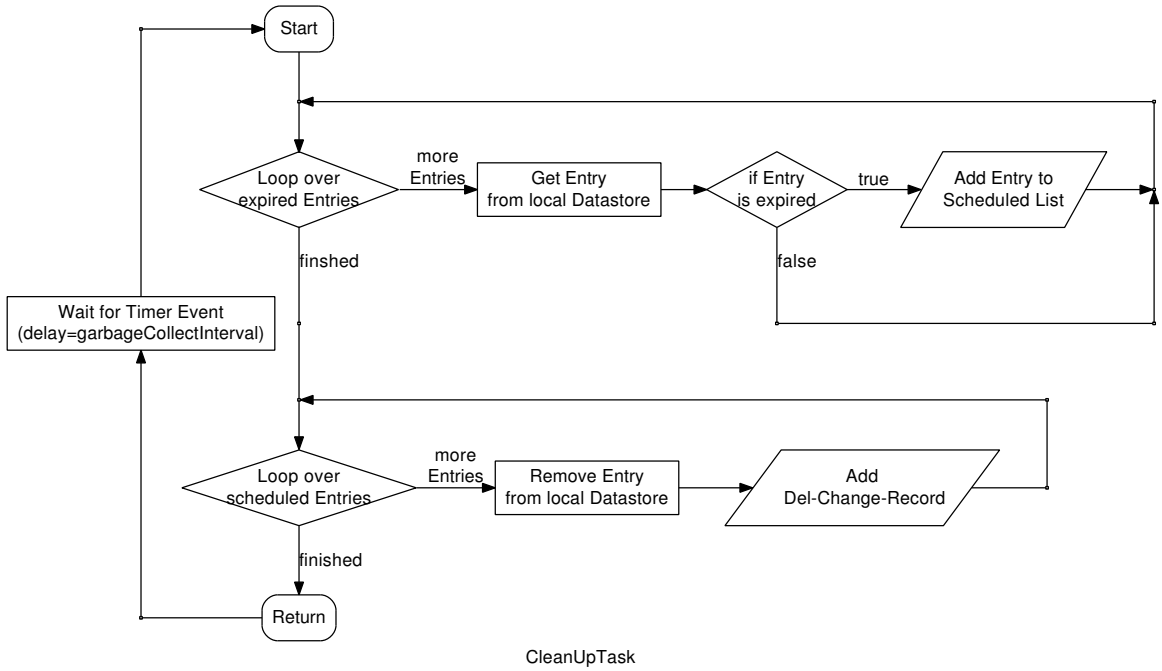


Abbildung 6.4: Die Logik der SRDI-Implementierung (Löschen veralteter Nutzdaten im Knoten)

6.3.2 Messungen am Prototyp

Die für die Messungen am Prototyp genutzte Umgebung basiert auf der Idee eines „virtuellen“ Netzes zwischen mehreren virtuellen Rechnern auf einem Host-System. Der Vorteil eines derartigen Aufbaus liegt in dem Fakt, dass nur wenig Hardware-Aufwand notwendig ist, um eine größere Menge an getrennt operierenden Applikationen zu erhalten. Da jedoch alle diese „virtuellen“ Rechner sich die Rechenleistung und den Hauptspeicher einer Hardware teilen, muss darauf geachtet werden, dass die Messwerte nicht durch eine zu hohe Last verfälscht werden. Versuche zeigten, dass es möglich ist, zwischen 5 und 10 virtuelle Rechner mit der Testapplikation auf einer Hardware-Instanz zu betreiben. Um den im folgenden Kapitel dargestellten Messaufbau und die Topologie des Netzes zu realisieren, wurde zur Messung des Kommunikationsaufwandes eine Wrapper-Bibliothek entwickelt und eingesetzt. Diese Bibliothek erlaubt es, alle kommunikationsrelevanten Funktionsaufrufe der Applikationen abzufangen und deren Daten zur Laufzeit der Applikation direkt in ein Logfile zu schreiben und sofort oder später aus-

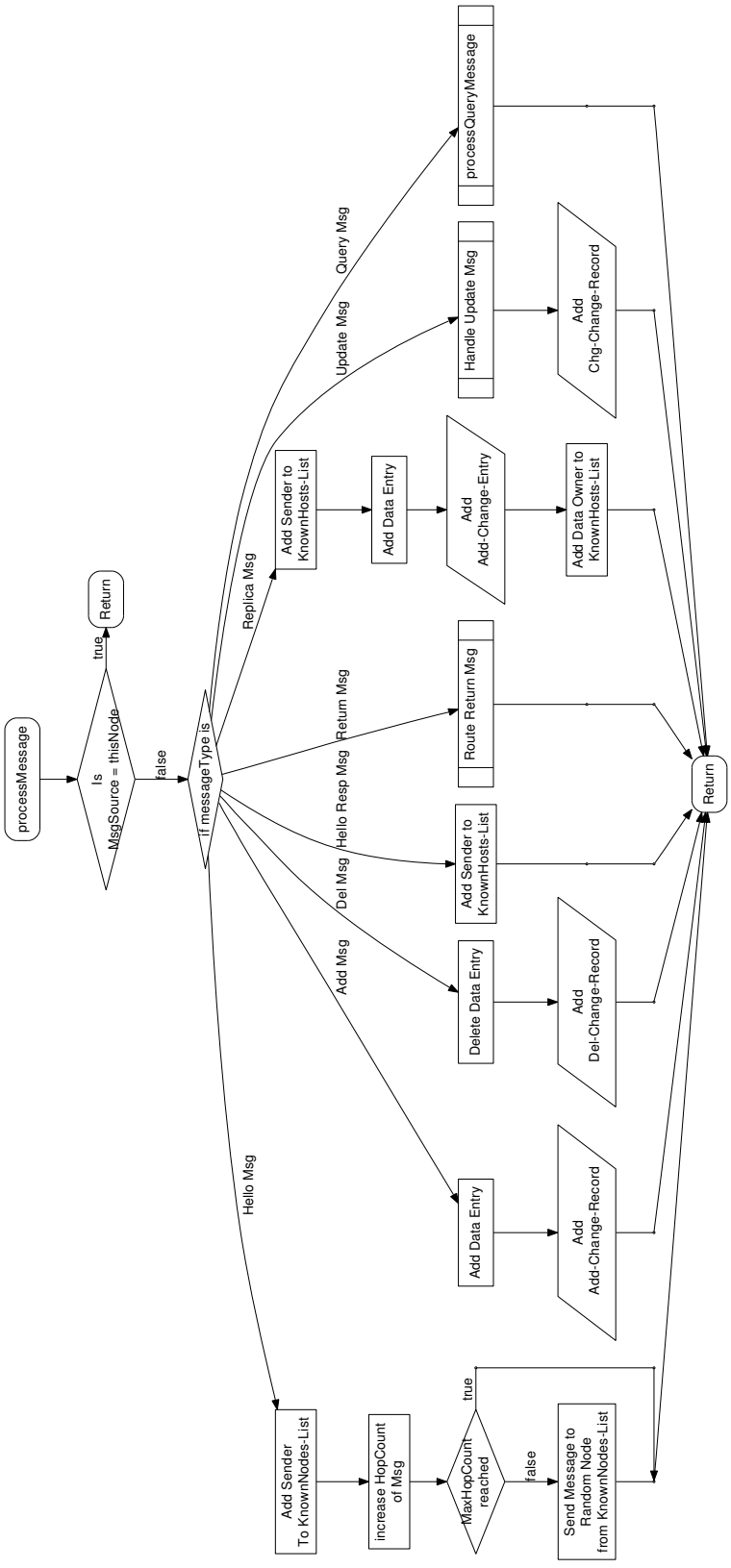


Abbildung 6.5: Die Logik der SRDI-Implementierung (Verarbeiten von eingehenden Paketen)

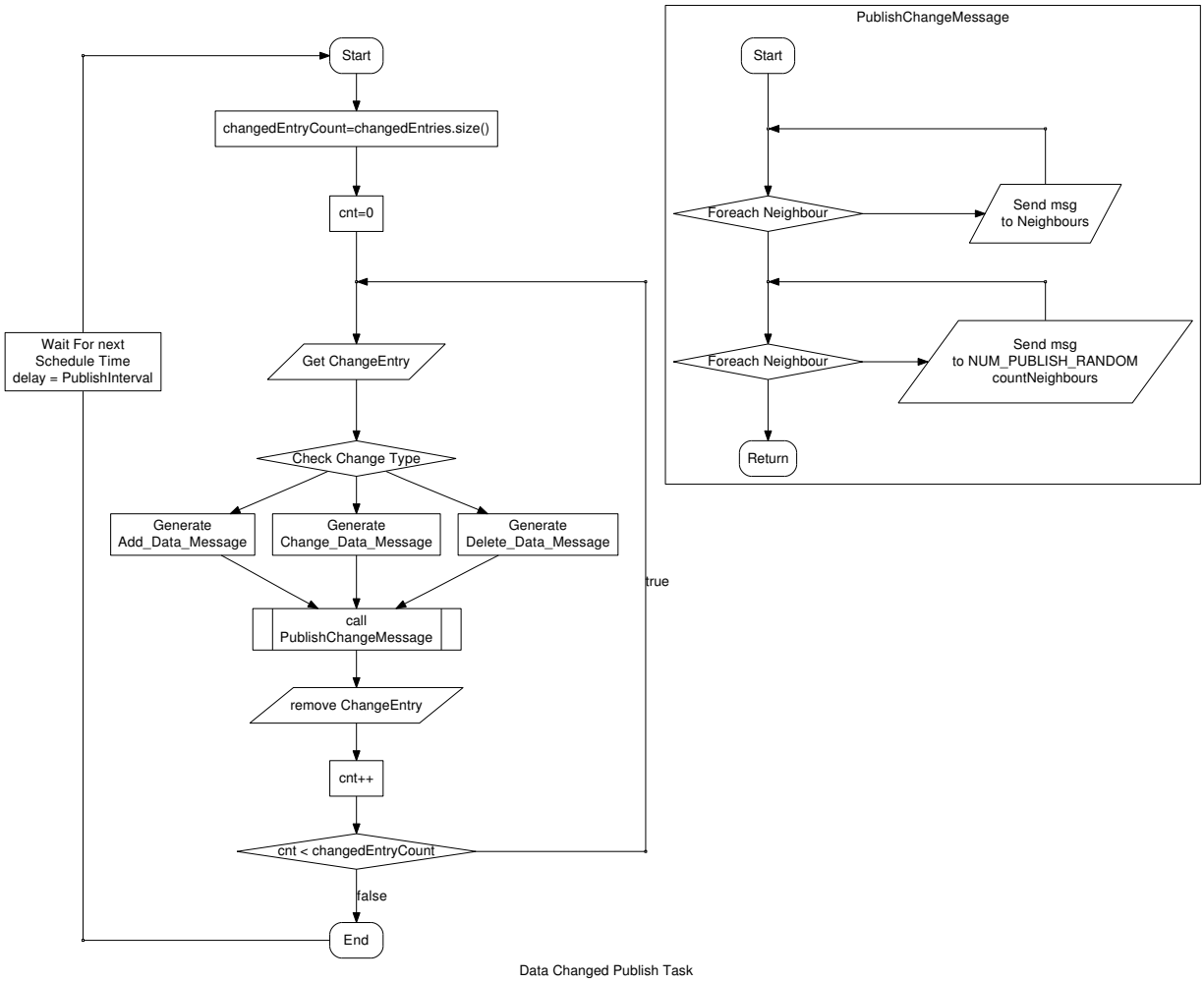


Abbildung 6.6: Die Logik der SRDI-Implementierung (Verteilen von Änderungen)

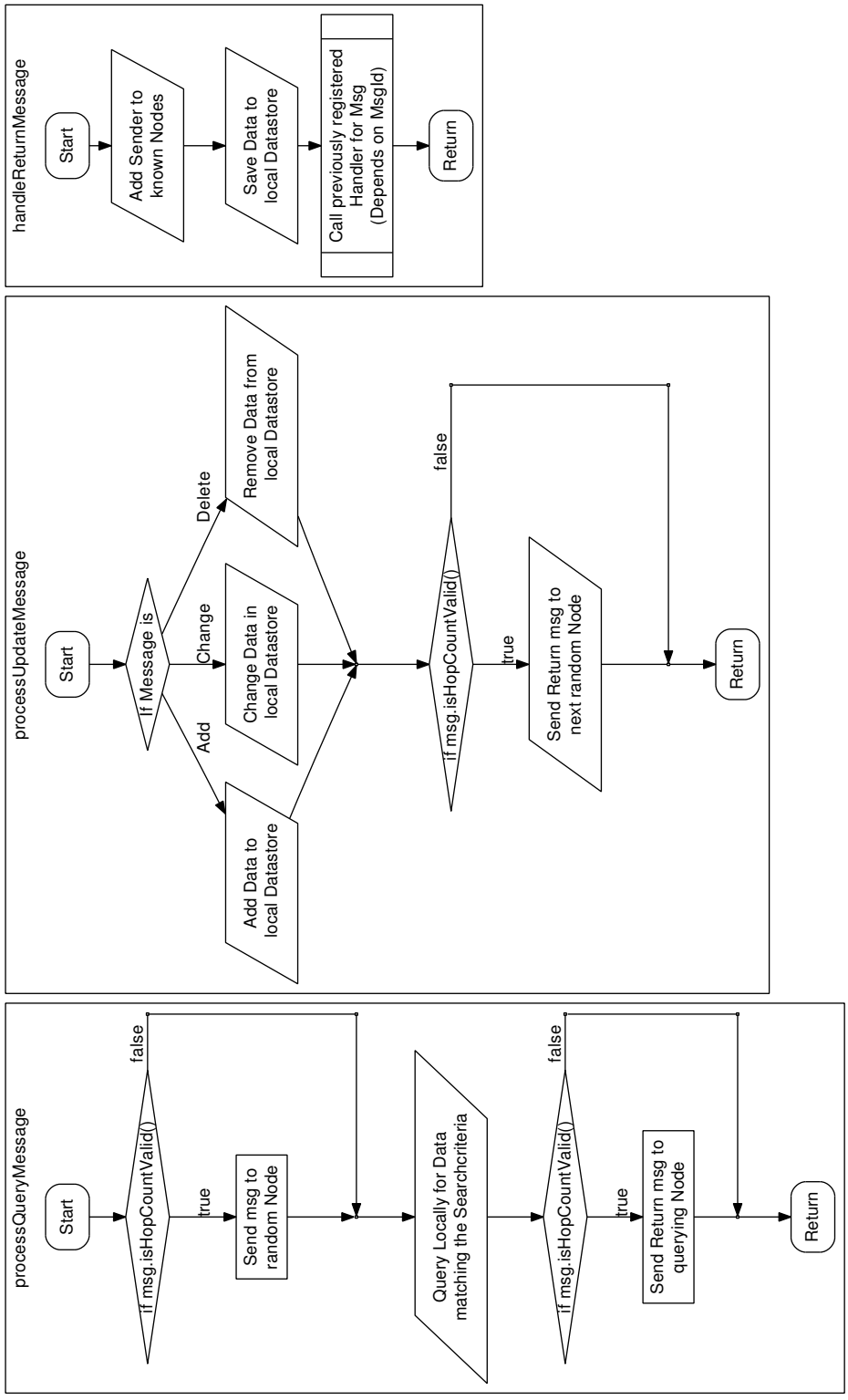


Abbildung 6.7: Die Logik der SRDI-Implementierung (Verschiedene Funktionen der Paketverarbeitung)

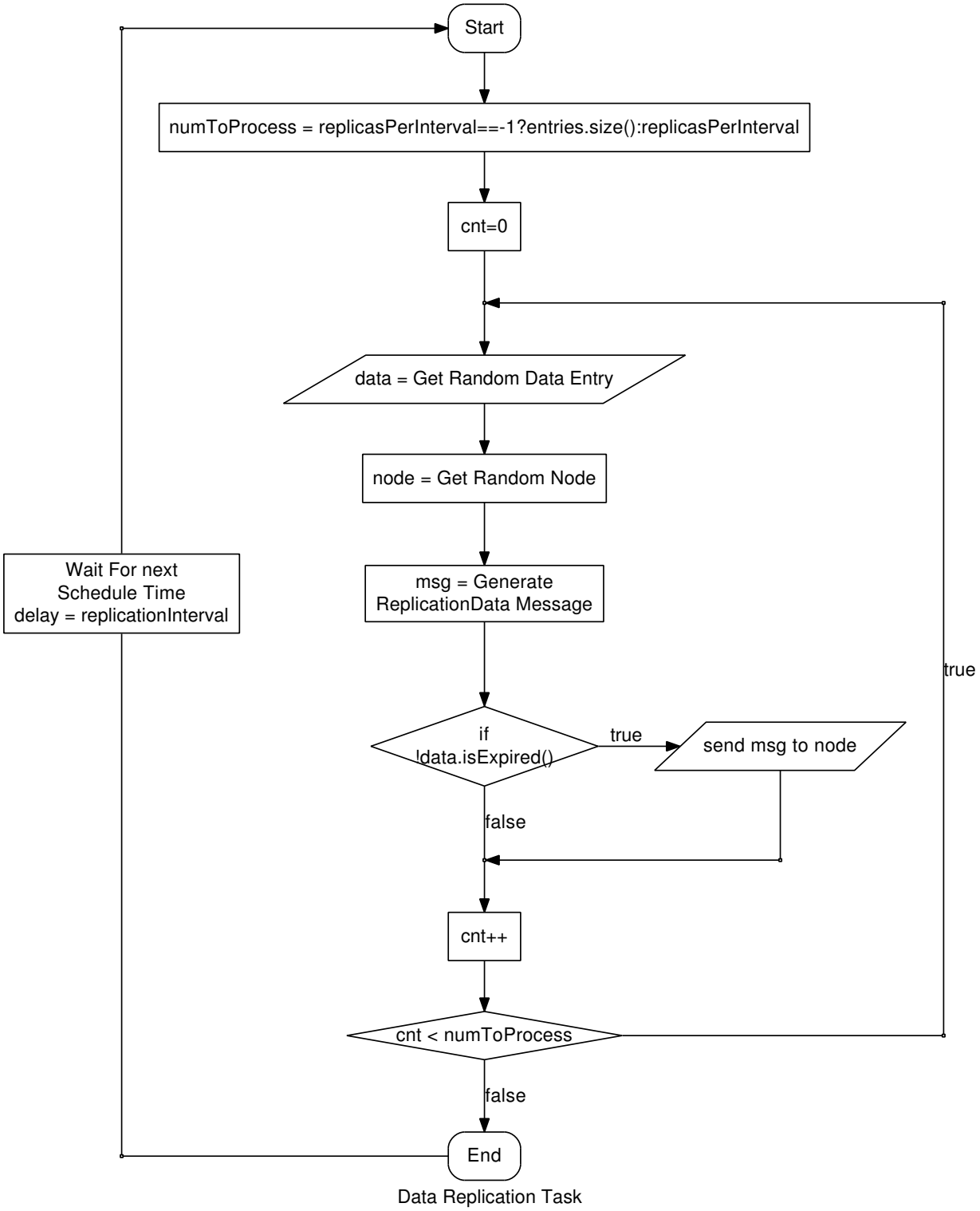


Abbildung 6.8: Die Logik der SRDI-Implementierung (Replizierung der Nutzdaten im Knoten)

```
protected  static int NUM_PUBLISH_RANDOM = 4;
public static int MAX_KNOWN_NODES_1 = 3;
protected  static int MAX_KNOWN_NODES_2 = 5;
protected  static int NUMBER_HELLO_NODES = 3;
protected  static int NUMBER_HELLO_MSG_HOPS = 2;

protected static int MAX_MSG_HOPS = 4;

protected static long localExpiaryTimeout = 10*60*1000;

private boolean shallPublishDirectly = false;
private boolean continueRunning = true;
private boolean REPLICA_TO_NEIGHBOURS_ONLY = false;
private boolean UPDATE_TO_NEIGHBOURS_ONLY = false;
```

Abbildung 6.9: Knotenparametrisierung der SRDI-Simulation

zuwerten. Der Vorteil dieses Ansatzes gegenüber der weit verbreiteten Nutzung von Werkzeugen wie tcpdump [19] oder ethereal [32] liegt in der wesentlich spezifischeren Erfassung der Daten. Es kann nicht nur erfasst werden, wann und welche Daten über das Netz transportiert werden, sondern es werden alle Kommunikationskanäle, die die Anwendung nutzt, überwacht und direkt an der Applikation erfasst. Die genaue Funktion und der Ablauf der Messungen wird in Anhang A beschrieben.

Aufbau der Messumgebung

Die Messungen wurden auf verschiedenen sich im LAN befindlichen "virtuellen Rechnern", auf welchen die P2P-Knoten verteilt wurden, durchgeführt. Zur Erfassung der benötigten Messwerte wurde eigens eine Messsoftware (siehe Anhang A) entwickelt, welche es erlaubt, ohne Änderung der Software eine genaue Erfassung der Datenstromparameter vorzunehmen.

```
<Simulation>
   <GeneralInformation>
      <Title>Dht Simulation</Title>
      <SimulationTool>fi.promotion.simpledht.StartTestNet</SimulationTool>
      <ResultsDirectory>/tmp/SimpleDht/Results_20061212</ResultsDirectory>
   </GeneralInformation>
   <ParameterSet>
      <FixedParameter Name="NumberOfBatches">10</FixedParameter>
      <FixedParameter Name="TransientTime">10000000</FixedParameter>
      <FixedParameter Name="TimePerBatch">10000000</FixedParameter>
      <FixedParameter Name="startupTime">2000000</FixedParameter>
      <FixedParameter Name="numReplicas">4</FixedParameter>
      <RangeParameter Name="nodes">
         <Value>1000</Value>
         <Value>100</Value>
      </RangeParameter>
      <RangeParameter Name="entries">
         <Value>10000</Value>
         <Value>150</Value>
      </RangeParameter>
      <RangeParameter Name="bandwidth">
         <Value>0.00375</Value>
         <Value>0.061</Value>
      </RangeParameter>
   </ParameterSet>
   <SimulationModel Name="TestNet">
      <ParameterSet>
         <FixedParameter Name="NumNodes">$nodes</FixedParameter>
         <FixedParameter Name="NUM_ENTRIES">$entries</FixedParameter>
      </ParameterSet>
      <Entity Name="Network">
         <ParameterSet>
            <FixedParameter Name="PacketDelay">100</FixedParameter>
            <FixedParameter Name="lineSpeed">
               $bandwidth
            </FixedParameter>
            <FixedParameter Name="writeDBMRStat"></FixedParameter>
         </ParameterSet>
      </Entity>
   </SimulationModel>
</Simulation>
```

Abbildung 6.10: Steuerdatei der SRDI-Simulation

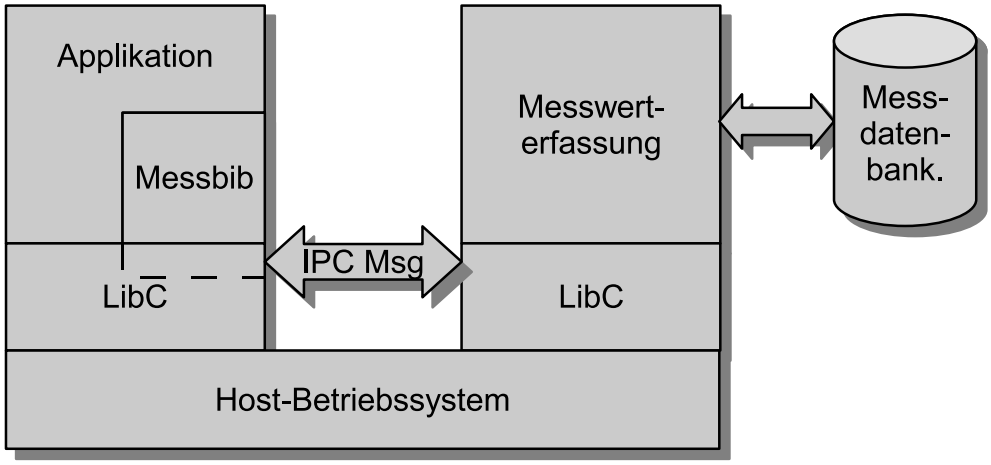


Abbildung 6.11: Konfiguration der Mess-Bibliothek

6.4 Ergebnisse aus Untersuchungen und Abschätzungen

Die folgenden Abschnitte beschreiben die Art der durchgeführten Untersuchungen, Überlegungen und Abschätzungen, sowie erste, darauf basierende, Ergebnisse.

6.4.1 Simulierte Teilbereiche

Um die Probleme der Skalierbarkeit und Stabilität des Netzwerk-Abstraktionslayers, das heißt des unterliegenden Peer-to-Peer-Systems, abschätzen und einordnen zu können, wurde im Rahmen der vorliegenden Arbeit, der dem System zugrunde liegende DHT-Mechanismus simuliert. Dieses ist, bezogen auf das Gesamtsystem, ein wichtiger Aspekt, da gerade zu dem Bereich des Management-Overhead des Peer-to-Peer-Netzes hierzu eine maßgebliche Aussage zu erwarten ist.

Ergebnisse der SRDI-Simulation

Im folgenden Kapitel wird auf die Ergebnisse einiger erster rudimentärer Simulationsläufe des oben beschriebenen Modells eingegangen.

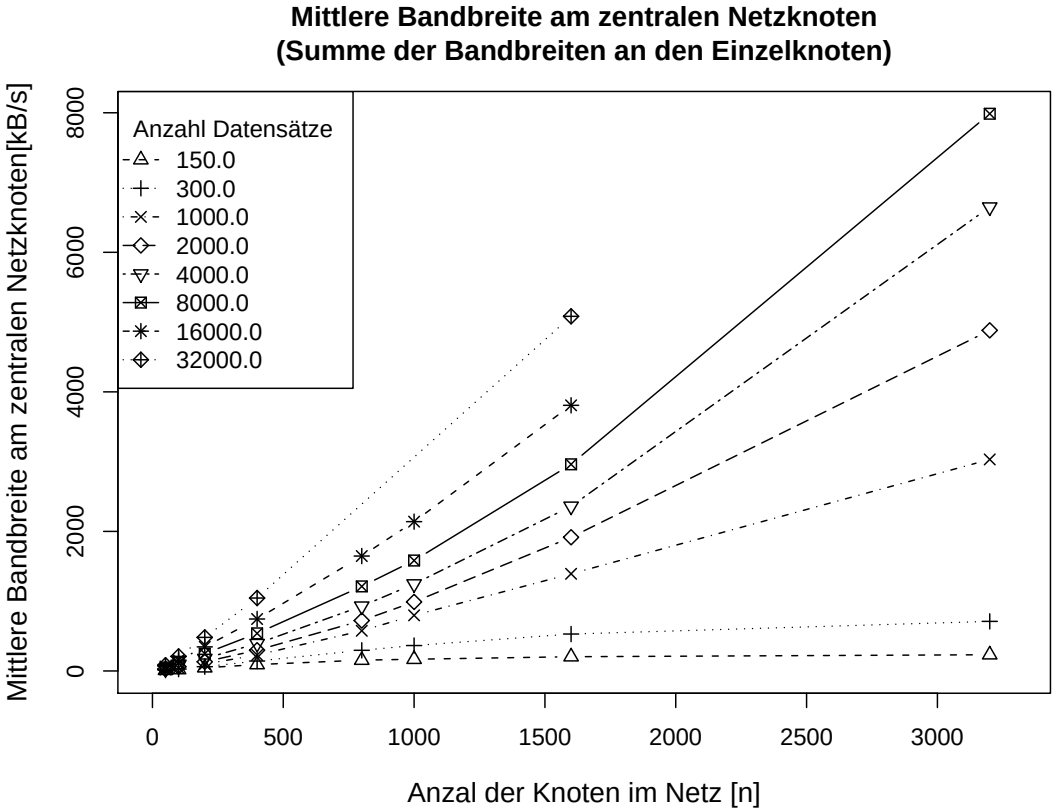


Abbildung 6.12: Bandbreite am Eingang des Netzes (Summe aller Knotenbandbreiten) in Abhängigkeit der Anzahl der Knoten im Netz und dem Parameter „Anzahl der Datensätze im Netz“

Wie Abb. 6.12 zeigt, steigt die durchschnittlich am zentralen Netzknoten genutzte Bandbreite sowohl mit der Anzahl der Knoten als auch, im Besonderen, mit der Anzahl der im Netz befindlichen Datensätze. Die durchschnittliche Bandbreite am zentralen Netzknoten ist hierbei die in der Simulation gemessene übertragene Datenmenge pro Zeiteinheit. Jedoch ist dabei die Variation der Bandbreite maßgeblich durch die Anzahl der Datensätze bestimmt und ändert sich mit steigender Anzahl der Knoten weniger stark als erwartet. Dies ergibt sich aus folgender Interpretation der Daten. Betrachtet man Abb. 6.13, so wird an der mittleren Bandbreite an den Einzelknoten deutlich, dass die Datenrate pro Knoten mit steigender Knotenzahl immer schwächer zunimmt. Ja im Falle von sehr wenigen Informationen bei vielen Knoten, sogar abnimmt. Ob hierbei generell ein asymptotisches Verhalten vorliegt, kann die im Folgenden kurz skizzierte Betrachtung

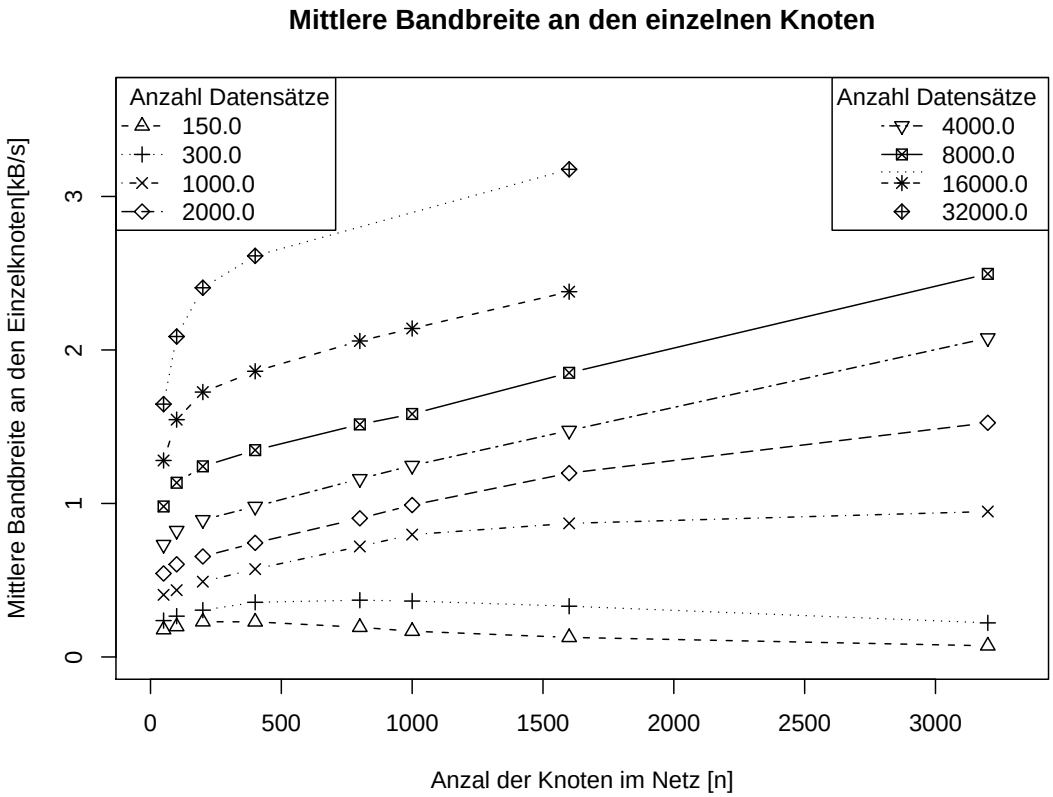


Abbildung 6.13: Durchschnittliche Bandbreite an den Netzknoten

zeigen.

Geht man davon aus, dass die genutzte DHT-Implementierung ab einer von ihrer Parametrisierung abhängigen Knotenmenge das Netz segmentiert, kann davon ausgegangen werden, dass die maximale Datenrate an den Einzelknoten durch die maximale Segmentgröße begrenzt ist, und sich damit ein asymptotisches Verhalten ergibt.

Mit zunehmender Anzahl von Datensätzen steigt die Variation der Bandbreite an den einzelnen Knoten jedoch erneut, so dass es hier zeitweise zu Blockaden des Systems in einzelnen Teilbereichen kommen kann.

Diese Blockaden wirken sich hierbei in einer Form aus, die es Institutionen mit niedriger Netzbandbreite unmöglich machen würde, das vorgeschlagene System einzusetzen.

Die Betrachtung des durch Knoten- oder Informationsfluktuation im Netz entstehenden zusätzlichen Bandbreitebedarfes durch den Managementoverhead des Peer-to-Peer-

Systems läßt sich am Verlauf der Datenrate an zufällig ausgewählten Knoten während des Starts des Systems unter Betrachtung des Kennwertes „[n/5]“ (Knotenanzahl im Netz zum Zeitpunkt t des Simulationslaufes) abschätzen. In den Abbildungen 6.14, 6.15 und 6.16 ist die Entwicklung der benötigten Managementbandbreite in Abhängigkeit der maximalen Knotenanzahl, der im System befindlichen Informationsmenge und der momentanen Anzahl an im Netz befindlichen Knoten aufgetragen und gegenüber gestellt.

Die eingehende Betrachtung zeigt deutlich, das der Managementoverhead maßgeblich durch die Kombination der zwei Paramater Knotenfluktuation und Inhaltsfluktuation bestimmt wird. Hierbei wird der Bandbreitebedarf des Gesamtsystems maßgeblich durch die pro Knoten fluktuierende Inhaltsmenge bestimmt, da deren „Verbreitung“ im Netz der Einzelknoten im Rahmen der gewählten DHT Implementierung dies impliziert.

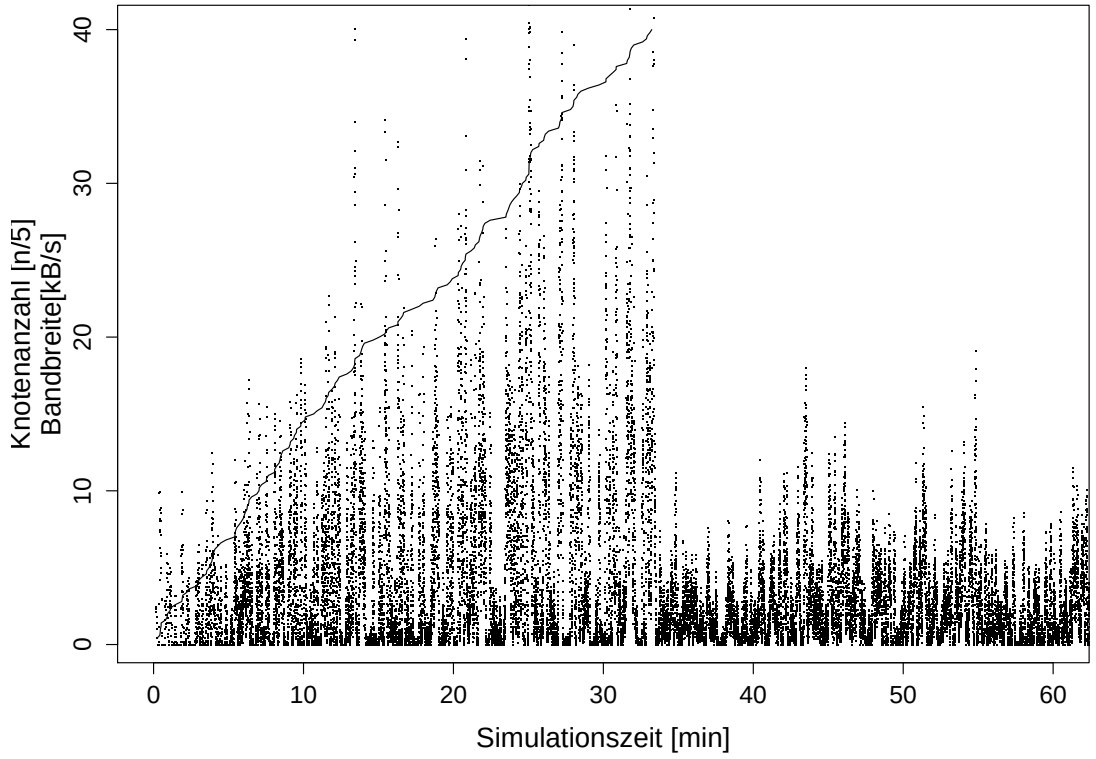


Abbildung 6.14: Entwicklung der Datenrate an zufällig ausgewählten Einzelknoten (200 Knoten, 1000 Einträge, Line-Rate 260kB/s) während des Systemstarts

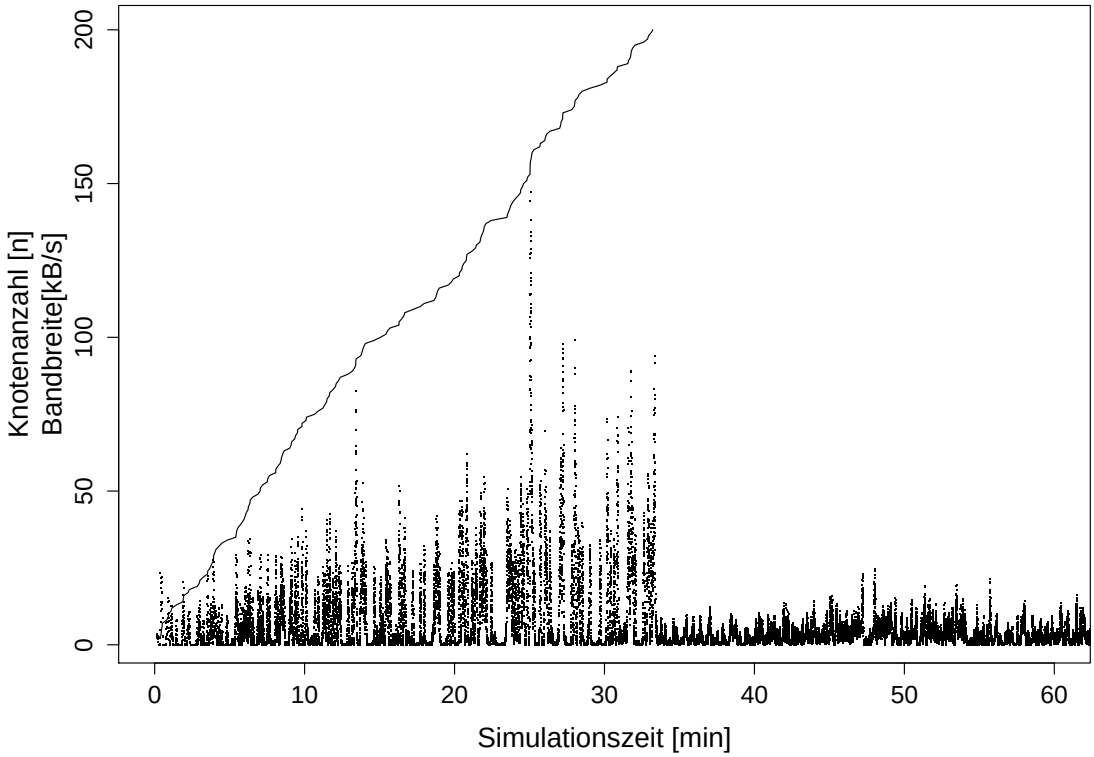


Abbildung 6.15: Entwicklung der Datenrate an zufällig ausgewählten Einzelknoten (200 Knoten, 2000 Einträge, Line-Rate 260kB/s) während des Systemstarts

6.4.2 Realisierte Teilbereiche

Um den Nachweis der prinzipiellen Machbarkeit zu führen und um die Kommunikationsaufwände des Gesamtsystems abschätzen zu können, wurden die Teilbereiche Inhaltserstellung und Autorensystem, ein rudimentärer System-Kern, zwei Transportmechanismen sowie der notwendige Teil des virtuellen Filesystems und einiger grundlegender Filesysteme realisiert. Wichtigste Filesysteme sind hierbei die verteilten Filesysteme sowie das Overlay-Filesystem. Darüber hinaus wurden in Ansätzen die für die angestrebte Bewertung notwendigen Schnittstellen und Hilfswerkzeuge für die Anbindung von beliebigen externen Applikationen in ihren Grundzügen implementiert.

Die folgenden Abschnitte führen in die einzelnen Teilbereiche und der Ergebnisse in der oben gewählten Reihenfolge ein.

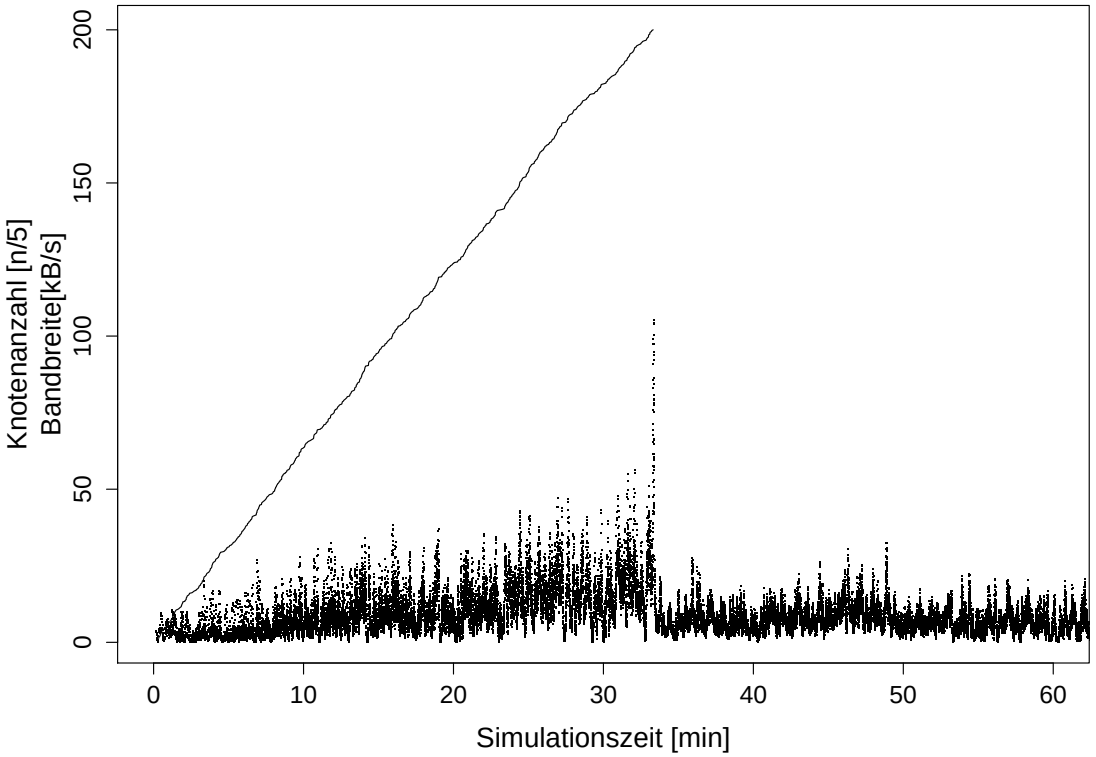


Abbildung 6.16: Entwicklung der Datenrate an zufällig ausgewählten Einzelknoten (1000 Knoten, 2000 Einträge, Line-Rate 260kB/s) während des Systemstarts

6.4.3 Abgeschätzte Teilbereiche

Die in den folgenden Abschnitten angegebenen Abschätzungen für die Teilbereiche Nutzer-Datenaufkommen, Metadaten-Aufkommen, Dokument-Datenaufkommen , Gesamtinformationsmenge und Teilnehmer-Anzahl im System beruhen entweder auf Erfahrungen und Daten aus den Projekten CANDLE und ITO, Statistiken von Forschungsinstituten und Behörden oder den Gegebenheiten des entworfenen Systems und seiner Eigenschaften.

Anhand der aufgelisteten Kennzahlen wird dann eine grobe Einordnung und Einschätzung des Systems abgegeben.

Abschätzung des Datenaufkommens durch die Nutzer

Zur Modellierung der durch die Nutzerinteraktion (Fortschreiten im Lernpfad, Annotationserstellung oder dem Nutzen anderer Anwendungen) entstehenden Datenströme müssen die in Abschnitt 3.2 beschriebenen Szenarien und die in Abschnitt 5.4.5 beschriebenen Mechanismen des Caching von Daten einbezogen werden. In die Modellierung fließen die im folgenden dargestellten Überlegungen zum zeitlichen und „räumlichen“ Verhalten der jeweilig betrachteten Nutzergruppe ein. Die prozentuale Zusammensetzung der Nutzergruppen ist auf eine universitäre Anwendung abgestimmt.

Größe der Metadaten

Die Bestimmung der durchschnittlichen Größe eines Metadatensatzes basiert auf Lehrmodulen und Kursen aus den Projekten ITO und CANDLE. Hierbei wurden insgesamt 24 Kurse untersucht, deren Struktur und Granularität variierten. Die Kurse stammen aus allen Bereichen der Informationstechnologie sowie aus verschiedenen Universitäten und Instituten aus Europa. Verschiedene Kurse aus der getroffenen Auswahl sind darüber hinaus noch in verschiedenen Granularitätsstufen bewertet worden. Damit stellen diese exemplarisch herangezogenen Kurse eine repräsentative Auswahl für den Bereich Informationstechnologie dar.

Die Auflistung der Kurse und ihrer Parameter findet sich in Anhang B.

Aus den Analysen ergibt sich eine mittlere Größe der Metadatensätze von

$$\gamma_{Metadaten} = 1,849 \pm 0,3 \text{ KByte}$$

Diese Größe geht später in die Betrachtung des durch Suche und Zugriff auf Kurse entstehenden Datenvolumens ein.

Dokumenten-Größen

Die in Abschnitt 4 beschriebene Abbildung unterschiedlicher Datenformate auf ein einheitliches Zwischenformat erlaubt eine Abschätzung der zu übertragenden Datenmenge für Kurse, Module und multimediale Elemente. Aus den exemplarisch aufbereiteten Vorlesungen VIS[3], CN I[4], EDECC[5] und 3D-Modelling ergibt sich, dass die Größenordnung für die jeweiligen Modul-Typen sich durchschnittlich im Bereich der in Tabelle 6.2 aufgelisteten Werte bewegen.

Tabelle 6.2: Übersicht über die Modulgrößen verschiedener Medientypen

Modul-Typ	Speicherformat	Größe	Anwendung
Text	XML	100 KByte	Lehrmodule
Bilder	SVG,JPEG,o.ä.	300 KByte	Grafiken in Lehrmodulen
3D	VRML	200 KByte	Spezielle Grafiken

Abschätzung der im System befindlichen Informationsmenge

Ausgehend von der in Abschnitt 5 beschriebenen Struktur des DLE-Systems kann die untere Grenze der im System befindlichen Informationsmenge in Abhängigkeit der Knoten und Diensteanzahl bestimmt werden. Die Berechnung basiert auf den Eigenschaften des JXTA-Systems und auf den vom DLE-System zusätzlich eingebrachten Eigenschaften.

Die Eigenschaften des JXTA führen auf folgende Informationsmengen:

Anzahl der Basis-Gruppen: Die Anzahl der Basisgruppen ergibt sich nach den Betrachtungen aus Abschnitt 5 direkt aus der Gesamtanzahl der am DLE partizipierenden Serversysteme. Die Klientensysteme spielen hier keine Rolle, da diese sich nicht direkt an den Basisgruppen sondern an einer privaten Gruppe des heimatlichen oder räumlich nächsten Serversystems anmeldet.

[3] Visualisierung
[4] Communication Networks I
[5] Error Detecting and Error Correcting Codes

Advertisements pro Gruppe: Die Anzahl der Advertisements pro Peer-Gruppe für einen Knoten ergibt sich aus der Summe der zum „Betrieb“ einer solchen Gruppe notwendigen Advertisments.

Die Aufgabe der Advertisements (Peer- oder Service-Advertisments) ist es unter anderem, die Zugehörigkeit eines Peers zu einer Peer-Gruppe zu beschreiben oder Metadaten von bereitgestellten Diensten bereit zu stellen. Als untere Grenze für die im System befindlichen Advertisements lässt sich die folgende Formel angeben:

$$\alpha_{gesamt}(\gamma) = \sum_{n=0}^{Anzahl\ der\ Gruppen} \alpha_n(\gamma)$$

wobei

γ : Anzahl der Systemgruppen

$\alpha_{gesamt}(x)$: Gesamtzahl der Advertisements in Abhängigkeit der Anzahl x der DLE-Server

$\alpha_n(x)$: Anzahl der Advertisements in Gruppe n in Abhängigkeit der Anzahl x der DLE-Server

welches sich für die untere Grenze berechnet als

$$\alpha_n(x) = \{\gamma * x + x\}$$

Abschätzung der Anzahl der am System teilnehmenden Institutionen und der Datenmengen

Für die Abschätzung der Anzahl der teilnehmenden Server-Knoten im DLE-System wird die Anzahl der in Deutschland existenten Hochschulen als Berechnungsbasis zugrundegelegt.

Tabelle 6.3: Institute an deutschen Hochschulen nach [44]

Hochschultyp	Anzahl	Fakultäten pro Hochschule	Professoren pro Hochschule	Studenten pro Hochschule
Universität	103	10-12	160	13400
Fachhochschulen	176	10-12	120	3200
Kunsthochschulen	53	3-5	20	600

Es gibt von den Vorreiter-Universitäten in Deutschland mittlerweile Aussagen über Nutzer und Datenmengen, die LM-Systeme nutzen und welche Datenmengen und Typen in sie eingestellt werden. Hierbei muss grundsätzlich zwischen den eigentlichen Basis-Lernmaterialien (Folien, Texten, Bildern, ...) und den Zusatz-Materialien (Vorlesungsaufzeichnungen) unterschieden werden. Es werden Zahlen von durchschnittlich 15.000 eingetragenen Nutzern genannt, wobei die Anzahl der aktiven Nutzer heute meist noch unter 1000 liegt.

6.4.4 Ergebnisse aus den Betrachtungen

Das vorgeschlagene System soll nun im Hinblick auf seine Akzeptanz bewertet werden. Die wesentlichen Parameter dieser Bewertung sind durch die Randbedingungen der Nutzbarkeit, des Inbetriebnahmeaufwandes, des Einarbeitungsaufwandes sowie der benötigten Bandbreite im Kommunikationsnetz bestimmt.

Netzlast-Abschätzung des vorgeschlagenen Systems

Betrachtet werden soll hier die zwischen den Server-Knoten und dem Weitverkehrsnetz benötigte Bandbreite, welche die existierenden Netze nur in geringem Maße belasten darf.

Die pro Server-Knoten benötigte Bandbreite kann hier mit Hilfe einer Datenflussanalyse aus den in Abschnitt 6.4.3 dargestellten Werten für die einzelnen Kommunikationsaufwände ermittelt werden.

Über diese Betrachtungen hinaus muss ein Maß für die im DLE-System notwendigen Relays gefunden werden. Die Anzahl und Positionierung von Relays ist abhängig von der Anzahl und Lokation durch Firewall- oder NAT[6]-Einrichtungen nicht direkt erreichbarer DLE-Server.

Die Eigenschaften von Relays legen nahe, dass die Bandbreite an diesen Knoten mindestens der Summe der Einzelknoten-Bandbreiten entspricht. Diese Betrachtungen haben jedoch nur bedingt Einfluss auf die Gesamt-Skalierbarkeit des DLE-Systems und können daher getrennt von den restlichen Betrachtungen durchgeführt werden.

Betrachtung der Ergebnisse des SRDI-Mechanismus

Für sich langsam ändernde Netze, das heißt Netze mit geringer Knoten- und Inhaltsdatenfluktuation, ist der SRDI-Mechanismus äußerst erfolgversprechend. Die Auswertung der Simulationsdaten belegt eindeutig, dass bei einer stabilen Menge teilnehmender Server-Knoten die benötigte Bandbreite zu vernachlässigen ist. Hohe Datenaufkommen werden nur bei hohen Fluktuationsraten von Teilnehmerknoten erzeugt.

Summenbildung der Einzelverkehre

Die beschriebene Struktur des DLE legt die im folgenden dargestellte Berechnung der Einzelverkehrssummen nahe. Jedes Serversystem verwaltet eine feste Anzahl von Grup-

[6] Network Address Translation

pen, jede Gruppe des Servers darüberhinaus die einschlägigen Dienste und die dazugehörigen Advertisements. Zu betrachten sind somit die Anzahl der Serversysteme, deren Gruppen und deren Dienste sowie die Anzahl der sich online befindenden Nutzer, sofern sich diese Anzahl auf den Kommunikationsaufwand einzelner Dienste auswirkt.

Verkehrsaufkommen pro Server-Knoten

Hier gibt es zwei unterschiedliche Ansätze:
Zum Einen die Berechnung der durchschnittlichen Belastung aus der in Abschnitt 6.4.3 deduzierten Gesamtrate, und zum Anderen, die Berechnung der durchschnittlichen Belastung für einen exemplarisch ausgewählten Dienstemix und eine durchschnittliche Benutzerzahl mit exemplarisch ausgewähltem Profil.

6.4.5 Aufwandsabschätzungen

Eine Aufwandsabschätzung kann hier nur auf Basis der Annahme, dass das vorgeschlagene System eine Basisnutzergemeinde, welche an der Weiterentwicklung mitarbeitet, hat, sinnvoll getroffen werden. Wird das System nur in Einzelfällen eingesetzt, so ist der Betriebs- und Nutzungsaufwand eher hoch. Nur die getroffene Annahme kann dafür sorgen, dass sich eine kritische Masse an Modulen zur Anbindung der verschiedensten Applikationen und Systeme bilden kann. Existieren diese Module, so kann ein neuer Nutzer sehr einfach und schnell das System seinen Anforderungen entsprechend in Betrieb nehmen.

Die Abschätzung zeigt, das es einen sehr hohen Anfangsaufwand zur Errichtung eines betriebsfähigen Systemes und der Erreichung der kritischen Masse an Modulen und Inhalten gibt. Für eine Nutzung eines so komplexes Systems nur in Einzelfällen oder in kleineren Insellösungen stellt es sich auf Grund des notwendigen Anfangsaufwandes als eher nicht praktikabel heraus. Geht man jedoch davon aus, das das in der vorliegenden Arbeit vorgestellte Problemfeld der modernen Lehre sich wie in Abschnitt 3 betrachtet entwickelt, und sich die Verwendung digitaler Medien, verteilter Labore und „weltwei-

ter“ Zusammenarbeit von Forschern und Studenten wie geschätzt entwickelt, so werden diese Aufwände ohnehin geleistet werden müssen. Dann erscheint es als sinnvoll den aus dem hier vorgeschlagenen System entstehenden Mehraufwand bei dessen Einführung auch zu leisten, um auf lange Sicht die damit zur Verfügung stehenden flexiblen Strukturen und Dienste gewinnbringend einsetzen zu können.

Kapitel 7

Zusammenfassung und Ausblick

Als Ausgangspunkt für die dargestellten Sachverhalte dient der vorliegenden Arbeit eine Betrachtung der heutigen E-Learning-Landschaft und ihrem Stellenwert für die weitere Entwicklung der durch moderne Medien unterstützten Lehre an Hochschulen in Deutschland und Europa. Mit der steigenden Verbreitung multimedialer Inhalte und der breiten Verfügbarkeit einer Internetanbindung der Lernenden ist die steigende Verbreitung von E-Learning nicht mehr aufzuhalten. Der Stand der Technik zeigt jedoch im realen Einsatz noch immer wesentliche Lücken an neuralgischen Punkten. Hier sind vor allem die Erstellung von Materialien und deren „Management“ über Institutionsgrenzen hinweg zu nennen. Learning-Management-Systeme (LMS), welche heute in der Breite Einsatz finden, haben eine zentralistische Struktur und keine oder nur rudimentäre Autorenwerkzeuge. Dies führt besonders bei Hochschulen, welche im Allgemeinen eine sehr heterogene Umgebung bilden, immer wieder zu Problemen, die eine breite Akzeptanz des jeweiligen LMS erschweren. Die Anforderungen, die von Außen gerade an die Hochschulen herangetragen werden, wie Austausch und Wiederverwendung von Materialien unter der Prämisse, bei allen Handlungen immer stärker auch wirtschaftliche Aspekte mit einzubeziehen, scheinen hier einen breiten Einsatz und verteilte Strukturen für LMS Lösungen zu fordern.

Viele Vorschläge und Systeme sind bereits entstanden, um Teilaspekte der beschriebenen Problematik zu lösen. Ein Konzept, das eine ganzheitliche Betrachtung und Lösung

angeht, stellt sich dagegen als wesentlich komplexer dar, als die Teillösungen vermuten lassen. Zwar existieren für fast alle Teilbereiche Standards, welche die Struktur, die Art und die Verknüpfung der Inhalte festlegen, jedoch scheint es hier immer eine Diskrepanz zwischen Komplexität und Anwendbarkeit zu geben. Das Ziel dieser Arbeit war es, eine geschlossene Lösung der wesentlichen Teilaspekte dieser Problematik zu präsentieren, welche unabhängig von dem bei der Institution eingesetzten LMS, einen dezentralen Austausch und Einsatz von Lehrinhalten erlaubt und den Anwendern dabei eine „leicht“ verwendbare Autorenumgebung bietet. Dadurch wurde das Ziel, die Akzeptanzschwelle für den E-Learning Einsatz auf ein Niveau zu senken, das den Einstieg wesentlich erleichtert, erreicht.

Es wurde hierzu in Kapitel 2 zunächst der technische Rahmen des Problemfeldes abgesteckt. Die wichtigsten Betrachtungsbereiche hier waren zum Einen die Randbedingungen, die die Struktur und Anforderungen der heute eingesetzten Transportnetze verteilten Systemen auferlegen, und zum Anderen die Anforderungen, die sich für einen nachhaltigen Einsatz von E-Learning-Materialien im Hinblick auf deren Wiederverwendung und Austausch etabliert haben. Besondere Beachtung lag auf einer ganzheitlichen Betrachtung aller Aspekte, woraus ein Profil der Anforderungen an ein Server-System zur Unterstützung von E-Learning Anwendungen abgeleitet wurde. Basierend auf diesem Profil wurde darauf aufbauend in den nachfolgenden Kapiteln die Architektur des Systems abgeleitet, und es wurden die zur Inhaltserstellung notwendigen Komponenten für Autorenwerkzeuge vorgestellt.

Eine weitere notwendige Betrachtung wurde in Kapitel 3 vorgestellt. Es wurde aufgezeigt und eingeordnet, welche nichttechnischen Randbedingungen eingehalten werden müssen, und welche wesentlichen Einsatzszenarien für ein solches Server-System zu betrachten sind. Es wurde verdeutlicht, dass es in dem betrachteten Umfeld wesentliche, nichttechnische, treibende Faktoren und randbedingungsgebende Einflüsse gibt. Es wurde herausgearbeitet, dass diese Einflüsse zum Teil ähnliche oder gleiche Auswirkungen haben, wie Teile der technisch bedingten Aspekte. Andererseits zeigte sich aber auch, dass „weiche“ Anforderungen zu harten Randbedingungen führen, welche für eine

Systemakzeptanz von hoher Bedeutung sind. Dies gilt im Besonderen auch und gerade für die Erstellung von E-Learning-Materialien, aber ebenso für dezentrale Aspekte des Managements der Plattform und der Nutzer.

Als Konsequenz der Problematik, ein akzeptables Werkzeug für die Inhaltserstellung bereitzustellen, wurde in Kapitel 4 eine mögliche Lösung aufgezeigt und ein dazu adäquates Werkzeug entworfen und beschrieben. Es wurde belegt, dass es, mit geringfügiger Steigerung der Komplexität für den Autor, möglich ist, verbreitete herkömmliche Autorensysteme und deren Möglichkeiten für die Erstellung standardkonformer Lehrmaterialien zu verwenden. Das entwickelte Autorenwerkzeug basiert auf einem Vorgehensschema, das hinreichend allgemeingültig ist, um für beliebige Autorenwerkzeuge anpassbar zu sein. Die Basisfunktionalität wurde an einer prototypischen Implementierung auf Basis des OpenOffice.org Paketes aufgezeigt.

Nachdem die Problematik der Inhaltserstellung verdeutlicht und ein Ansatz zur Lösung präsentiert wurde, muss nun nachfolgend die Aufgabe der verteilten Verwendung der erstellten Materialien betrachtet werden. Kapitel 5 enthält hierzu einen Architekturvorschlag, der die erarbeiteten Anforderungen und Randbedingungen erfüllt. Hierzu wurden bekannte Mechanismen verteilter Systeme mit Sicherheitsmechanismen und einem Overlay-Netz kombiniert. Die Architektur ist hierbei so beschaffen, dass sie selbst auch als Zwischenschicht, das heißt, als sicheres, leicht installierbares und verteilt administriertes Transport-, Verteil- und Zugriffssystem auf E-Learning-Anwendungen unterstützende Dienste und Daten fungieren kann.

Die notwendigen Funktionalitäten und Mechanismen wurden, um eine Bewertung der Architektur zur ermöglichen, in Teilen implementiert. Die Implementierung und ihre Eigenschaften sind in Kapitel 6 beschrieben. Die Implementierung ist als Proof-of-Concept angelegt und wird einem Produktionssystem sicher nicht gerecht. Sie ist jedoch ausreichend, um eine funktionelle Prüfung der Konzepte und der durch sie entstehenden Datenaufwände im Netz zu gewährleisten. Hierbei ist die Implementierung selbst an keiner Stelle optimiert, es werden jedoch die problematischen Bereiche angesprochen und etwaige Lösungsansätze angegeben.

Auf Basis der Implementierung und einer Simulationsstudie zum Verhalten des DHT-Mechanismus des unterliegenden Peer-to-Peer-Overlay-Netzes, sowie anhand der erarbeiteten Randbedingungen werden die für eine weitergehende Analyse der Skalierbarkeit und der Akzeptanz der vorgestellten Architektur notwendigen Basisansätze angegeben und ausgeführt. Anhand dieser Ansätze wurde eine rudimentäre Bewertung vorgenommen und die Auswirkungen eines solchen Systems, ausgehend vom momentanen Status der Bildungslandschaft, abgeschätzt. Die gegebene Abschätzung zielt darauf, die wesentlichen Vor- und Nachteile der Technologie und ihrer Auswirkungen im Bezug auf die universitäre Lehre zu beleuchten. Es wird erwartet, dass die Aussage, dass ein solches, unter den Randbedingungen der Akzeptanz und Skalierbarkeit durchaus zu realisierendes System, die Verbreitung des Einsatzes von vorlesungsunterstützenden E-Learning Angeboten fördert und wesentlich dazu beiträgt, einen nachhaltigen Einsatz von E-Learning sicherzustellen, verifiziert wird.

Die vorliegende Arbeit zeigt auf, dass das vorgeschlagene System so beschaffen ist, dass es weitest möglich von Betriebssystem und Netztechnologie beziehungsweise der Netzanbindung und den daraus entstehenden Randbedingungen unabhängig dezentral installiert und genutzt werden kann. Die einzelnen Systeme bilden ein Netz dezentraler Server und ermöglichen einen verteilten Zugriff auf die Ressourcen der Einzelsysteme. Das die Interaktion sowie den Datenaustausch der Server unter der Berücksichtigung mobiler Nutzer und deren Applikationen netztransparent ermöglicht, und dass diese Netztransparenz erreicht wird, ohne dass zum Beispiel Sicherheitseinrichtungen wie Firewalls oder Netzwerk Adress Translators im Netz der jeweiligen Institution speziell konfiguriert sein müssen. Sowie, dass sich die auf JXTA basierende Transportschicht des vorgeschlagenen Systems für die angestrebte Anwendung auch unter den gegebenen Skalierbarkeitsanforderungen bewährt hat.

Die Betrachtung zeigt aber auch, dass es noch weiterer und weitergehender Standardisierungen in vielen der angesprochenen Bereiche der Inhaltserstellung, -verwaltung und deren Beschreibung mit Metadaten bedarf. Dies gilt im Besonderen für die Unterstützung der Autoren bei der Inhaltserstellung und dem Austausch dieser Inhalte über

die verschiedenen existierenden LMS- und CMS-Systeme hinweg. Darüber hinaus bleibt die Problematik der Strukturierung und Wiederverwendbarkeit und der dazu notwendigen Metadaten und deren Inhalte für die erstellten Lernmodule ein wesentlicher Beriech der für die Forschung zu lösen bleibt. Hier zeigt die vorliegende Arbeit zum Beispiel für die Frage nach einer allumfassenden Strukturierungsmethode für die Suche nach E-Learning-Modulen zwar Ansätze auf, kann jedoch keine allgemeingültige Lösung anbieten oder bewerten. Die angegebenen Strukturierungsmethoden stellen wichtige Ansätze dar um das angesprochene Problem zu lösen, sie sind von einer generalisierten Lösung jedoch noch ein gutes Stück entfernt ist. Diese generalisierte Lösung und eine genaue Analyse ihre Notwendigkeit sowie eine umfassende Studie zu Ihrer Machbarkeit sind sicher spannende Forschungsaufgaben in diesem Bereich.

Dies gilt zusammenfassend auch für die in der vorliegenden Arbeit erreichten Ergebnisse, welche im Folgenden noch einmal kurz zusammengefasst bewertet und zu weiteren Forschungen und Studien zugeordnet werden sollen.

Randbedingungen für den erfolgreichen und nachhaltigen Einsatz von E-Learning:

Die erarbeiteten Randbedingungen stellen die Grundlage für einen erfolgreichen und nachhaltigen Einsatz von E-Learning in der Lehre dar. Die wichtigen Erkenntnisse sind hierbei die Komplexität und die Mannigfaltigkeit der Abhängigkeiten die sich für das gestellte Ziel ergeben.

Erstellen von Lehrmaterialien mit herkömmlichen Autorensystemen und deren Transformation in ein systemunabhängiges Zwischenformat:

Das Ergebnis der vorliegenden Arbeit hier zeigt auf, wie die Randbedingungen im Bereich der Inhaltserstelllung erfüllt werden können. Das vorgestellte Konzept einer in ein generelles Autorensystem eingearbeiteten und durch komplexe Transformationsmechanismen ergänzten Softwarekomponente zeigt dieses auf. Es zeigt aber auch, das die vorgeschlagene Lösung alleine für die verbreitete Akzeptanz des wenn auch verminderten Mehraufwands für die Autoren nicht ausreicht. Hier muss in den Köpfen der Autoren ein Umdenkprozess in Gang gesetzt werden, der sie den Mehrwert deutlicher erkennen lässt. In diesem Bereich ist die Forschung gefordert, zum Beispiel im Bereich der

Definition von Randbedingungen an zu erstellendes Material zum Ziel der einfacheren Wiederverwendbarkeit zu erarbeiten wie dieses Beschrieben werden muss.

Erarbeiten einer Architektur für ein verteiltes P2P-Basiertes System das die Randbedingungen erfüllt:

Die vorliegende Arbeit stellt ein neues Architekturkonzept vor, das die gegebenen Randbedingungen mit Peer-to-Peer Konzepten erfüllt. Hierbei werden alle erarbeiteten Anforderungen in die Konzeption einbezogen und die notwendigen Architekturkomponenten definiert. Die Mannigfaltigkeit der benötigten Komponenten macht eine tiefgehende Bewertung hier jedoch nur schwer möglich. Es werden aber viele Ansätze aufgezeigt die einer vertiefte Betrachtung lohnend erscheinen lassen. Zu nennen wären hier zum Beispiel die A4C Umgebung oder die Problematik der Remote-Code-Verification und des sicheren verteilten Cachings von der Inhaltsdaten.

Prototypische Implementierung und Bewertung der wesentlichen Architektur-Bestandteile:

Die im Rahmen der Arbeit gemachten prototypischen Implementierungen stellen einen Proof-of-Technology dar. Dies ist Ausreichend um wesentlichen Aufschluss darüber zu geben, wie die gestellten Randbedingungen zu erfüllen sind, respektive wie sie erfüllt werden können. Die Implementierungen zeigen hierbei auf, wie die Bereiche A4C, Inhaltsverteilung, Personalisierung und Chaching auf Basis des Peer-to-Peer Paradigmas zu lösen sind. Hier bietet die vorliegende Arbeit eine sehr gute Basis für die Realisierung eines Prototyps an welchem sich dann weitere Untersuchungen durchführen ließen, welche die folgenden beleuchteten Simulationsergebnisse ergänzen würden.

Simulation des P2P-Systems zur Untersuchung der Einsatzfähigkeit der vorgeschlagenen Architektur:

Die simulative Bewertung des der vorliegenden Arbeit zugrunde liegen DHT Mechanismus unter Berücksichtigung der Architektur des Gesamtsystems gibt Aufschluss über die Eignung des vorgeschlagenen Systems. Die Betrachtung zeigt, das der Einsatz prinzipiell möglich ist und die Architektur des Systems gut gewählt ist. Es wird aber auch deutlich, das die Komplexiät eines solch umfassenden Systems weitere Studien über sein Verhalten notwendig macht. Das im Rahmen der Arbeit entwickelte und erarbeitet

Simulations- und Messsystem bietet hier sehr gute Ansätze dies zu tun. Im Besonderen wären weitergehende Studien im Bereich der Simmulation des P2P-Systems, zum Beispiel bezogen auf das Verhalten bei der Suche oder Knoten und Inhaltsfluktuationen lohnenswerte Bereiche weiterer Forschungen.

Literaturverzeichnis

[1] Tom Anderson, Michal Dahlin, Jeanna Neefe, Drew Roselli, David Patterson, and Randy Wang. Serverless network file systems. In *ACM SOSP*. ACM SOSP, December 1995.

[2] bamboo dht.org. The bamboo distributed hash table: A robust, open-source dht. `http://www.bamboo-dht.org/`, March 2005. Web site introducing a Bandwidth efficent DHT structure based on the Pastry metric.

[3] David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Christopher Wells, Ben Zhao, , and John Kubiatowicz. Oceanstore: An extremely wide-area storage system. Technical Report UCB/CSD-00-1102, University of California, 1999.

[4] Andreas Binzenhöfer and Phouc Tran-Gia. Delay analysis of a chord-based peer-to-peer file-sharing-system. Technical report, University of Würzburg Institute of Computer Science, 2004.

[5] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508`, May 2000. W3C Note.

[6] Peter J. Braam. The coda distributed file system. *Linux Journal*, 1998(50es), 1998.

[7] Randy Brown. Calendar queues: A fast 0(1) priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, October 1988.

[8] Brent Callaghan, Brian Pawlowski, and Peter Staubach. Nfs networking file system protocol specification (rfc 1813). RFC 1813, IETF, June 1995.

[9] Kathryn Chen, Loai Naamani, and Karim Yehia. michord: Decoupling object lookup from placement in dht-based overlays. Technical report, Massachusetts Institute of Technology, 2003.

[10] SWAP Consortium. Swap - semantic web and peer-to-peer. `http://swap.semanticweb.org/public/index.htm`.

[11] Gennaro Cordasco, Vittorio Scarano, and Cristiano Vitolo. Architecture of a p2p distributed adaptive directory. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 282–283, New York, NY, USA, 2004. ACM Press.

[12] Arturo Crespo and Hector G. Molina. Semantic overlay networks for p2p systems, 2002.

[13] Arturo Crespo and Hector G. Molina. Semantic overlay networks for p2p systems. Technical report, Computer Science Department, Stanford University, 2002.

[14] DARPA. Daml - the darpa agent markup language homepage. `http://www.daml.org`, May 2005.

[15] Edutella Developers. Edutella - p2p for the semantic web. `www.edutella.org`, July 2004.

[16] Tim Dierks and Christopher Allen. The tls protocol version 1.0. Request For Comments (RFC-2246), January 1999. RFC-2246.

[17] Theresa Edgington, Beomjin Choi, Katherine Henson, T. Raghu, and Ajay Vinze. Adopting ontology to facilitate knowledge sharing. *Commun. ACM*, 47(11):85–90, November 2004.

[18] Oliver Meier et. al. Basics of information-broker architecture. Technical report, CANDLE-Consortium (IRA-UKA), 2000.

[19] Bill Fenner. Tcpdump homepage. `http://www.tcpdump.org`, 2007.

[20] Lutz Finsterle and Martin Rotard. Mit konventionellen Autorensystemen zum E-Learning Portal. In *Von e-Learning bis e-Payment, Das Internet als sicherer Marktplatz, Tagungsband Lit '02*, pages 111–121, September 2002.

[21] Luis Garces-Erice, Ernst W. Biersack, Keith W. Ross, Pascal A. Felber, and Guillaume Urvoy-Keller. Hierarchical peer-to-peer systems. In *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003.

[22] GartnerGroup. The emergence of distributed content management and peer-to-peer content networks. Technical Report Engagement No. 010022501, GartnerGroup, January 2001.

[23] Li Gong. Jxta: a network programming environment. *Internet Computing, IEEE*, 5(3):88–95, 2001.

[24] Li Gong. Project jxta: A technology overview. *WWW*, -(-):1–12, October 2002. Overview of JXTA Technology and some interesting numbers of internet development inclusive some good figures of the software architecture.

[25] Gerhard Haßlinger, Kurt Tutschku, and Phuoc Tran-Gia. *Peer-to-Peer Systems and Applications*, chapter Peer-to-Peer Traffic and Performance Evaluation (Part VII.), pages 369–397. Lecture Notes in Computer Science. Springer, 2005.

[26] Sabine Henneberger, Matthias Schulz, and Jakob Voss. Save as xdiml, writing and converting digital theses and dissertations using openoffice.org. In *Proceedings of the OpenOffice.org Conference*. OpenOffice.org, 2003.

[27] The Dublin Core Metadata Initiative. The dublin core webpages. http://www.dublincore.org, 2005.

[28] Gilles Kister. Modellierung und Leistungsuntersuchung von Peer-to-Peer Netzen zur unstrukturierten Suche in heterogenen Umgebungen. Diplomarbeit, University of Stuttgart, November 2004.

[29] John Kohl and B. Clifford Neuman. The kerberos network authentication service (version 5). Internet Request for Comments RFC-1510, September 1993. RFC-1510.

[30] DCE Konsortium. Dce/dfs standard. http://www.opengroup.org/dce, 1997.

[31] Rob Koper. Modeling units of study from a pedagogical perspective. *EML*, 1(1):1–40, June 2001. Draft Version 2.

[32] Ulf Lamping, Richard Sharpe, and Ed Warnicke. Ethereal user's guide. http://www.ethereal.com/docs/eug_html, 2005. Ethereal Web-Page.

[33] Ora Lassila and Ralph R. Swick. Resource description framework (rdf) model and syntax specification.

[34] Jinyang Li, Jeremy Stribling, Robert Morris, M. Frans Kaashoek, and Thomer M. Gil. A performance vs. cost framework for evaluating dht design tradeoffs under churn. Technical report, Massachusetts Institute of Technology - Computer Science and Artificial Intelligence Laboratory, 2004.

[35] David Liben-Nowell, Hari Balakrishnan, and David Karger. Analysis of the evolution of peer-to-peer systems. Technical report, Massachusetts Institute of Technology, 2002.

[36] Lom - learning objects metadata. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf.

[37] Eetu Mäkelä, Eero Hyvönen, and Samppa Saarela. Ontogator - a semantic view-based search engine service for web applications. In *International Semantic Web Conference*, pages 847–860, 2006.

[38] Andreas Mauthe and David Hutchison. Peer-to-peer computing: Systems, concepts and characteristics. *PIK*, 26(2):60–64, 2003.

[39] Metacoon - free software for e-learning, e-authoring, e-community, e-work, e-knowledge, e-government. `http://www.metacoon.net/`, 2004.

[40] Microsoft. Introduction to windows peer-to-peer networking. Technical report, Microsoft, 2003.

[41] Peter Mika and Hans Akkermans. Analysis of the state-of-the-art in ontology-based knowledge management. Technical report, Division of Mathematics and Informatics, Vrije Universiteit, Amsterdam, 2003.

[42] Definition of software and hardware-modules in websters dictionary. `http://www.webster-dictionary.org/definition/module`, October 2004.

[43] Wiebke Oeltjen. Openoffice.org as an authoring tool in the figaro project. In *Proceedings of the OpenOffice.org Conference*, 2003.

[44] Federal Ministry of Education and Science, editors. *Education Statistics for the Federal Republic of Germany*. Basic and Structural Data. Federal Ministry of Education and Science, Heinemannstraße 2, Referat I B 6, 5300 Bonn 2, 1991/1992 edition, 11 1991.

[45] Massachusetts Institute of Technology Kerberos Team. Kerberos: The network authentication protocol. `http://web.mit.edu/kerberos/www/`.

[46] Pinar Ozturk and Rolv Break. Creem - content reengineering manual. Manual WP5-NTNU-03, CANDLE, 2001. (Not officially avaliable).

[47] Göhner P., editor. *Simulationen und Animationen mit Java-Applets*, chapter 4.2, pages 100–111. Waxmann Verlag, 2004.

[48] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl - web ontology language. http://www.w3.org/TR/owl-semantics/, Febuary 2004.

[49] Andy Powell, Mikael Nilsson, Ambjörn Naeve, Pete Johnston, and Thomas Baker. Dcmi abstract model. http://dublincore.org/documents/abstract-model/, March 2005.

[50] Andrew P. Rifkin, Michael P. Forbes, Richard L. Hamilton, Michael Sabrio, Suryakanta Shah, and Kang Yueh. Rfs architectural overview. In *USENIX Summer Conference Proceedings*, pages 248–259, 1986.

[51] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[52] Demetrios G. Sampson, Miltiadis D. Lytras, Gerd Wagner, and Paloma Diaz, editors. *Special Issue on: Ontologies and the Semantic Web for E-learning*, volume 7. International Forum of Educational Technology & Society, 2004.

[53] ISO JTC1 / SC34. Xml topic maps. http://topicmaps.org/, Febuary 2001.

[54] Eran Segal, Dana Pe'er, Aviv Regev, Daphne Koller, and Nir Friedman. Learning module networks. Technical report, Stanford University, 2003.

[55] Spencer Shepler, Brent Callaghan, David Robinson, Robert Thurlow, Carl Beame, Mike Eisler, and David Noveck. Nfs v4 networking file system protocol specification (rfc 3010). RFC 3010, IETF, December 2000.

[56] Spencer Shepler, Brent Callaghan, David Robinson, Robert Thurlow, Carl Beame, Mike Eisler, and David Noveck. Nfs v4 networking file system protocol specification (rfc 3530). RFC 3530, IETF, April 2003.

[57] Weidong Shi, Hsien-Hsin S. Lee, Mrinmoy Ghosh, and Chenghuai Lu. Architectural support for high speed protection of memory integrity and confidentiality in multiprocessor systems. In *PACT '04: Proceedings of the 13th International Conference on*

Parallel Architectures and Compilation Techniques, pages 123–134, Washington, DC, USA, 2004. IEEE Computer Society.

[58] Scaleability of p2p network models. WWW.

[59] Ralf Steinmetz and Jörg Eberspächer. Editorial peer-to-peer. *PIK*, 26(2):59, 2003.

[60] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Springer, October 2005.

[61] Inc. Sun Microsystems. Rpc: Remote procedure call protocol specification version 2(rfc 1057). RFC 1057, IETF, June 1988.

[62] Inc. Sun Microsystems. Nfs v2 networking file system protocol specification (rfc 1094). RFC 1094, IETF, March 1989.

[63] Sun.Com. Dynamic web-pages by using applets. `http://java.sun.com/applets`.

[64] Tapestry a new dht-schema. `http://current.cs.ucsb.edu/projects/chimera/`, May 2005.

[65] ADL Technical Team. Sharable content object reference model (scorm) version 1.2. `http://www.adlnet.gov/downloads/AuthNotReqd.aspx?FileName=SCORM_1_2_pdf.zip&ID=240`, January 2001.

[66] Bernard Traversat and Ahkil Arora Mohamed Abdelaziz. Project jxta 2.0 super-peer virtual network. *http://www.jxta.org/*, 1(1):1–20, May 2003.

[67] Bernard Traversat, Mohamed Abdelaziz, and Eric Pouyoul. Project jxta: A loosely-consistent dht rendezvous walker. Technical report, Sun Microsystems, 2003.

[68] Thomas Veigel. Entwurf einer AAA-Umgebung für ein auf JXTA-basierdendes verteiltes Lehrmodul Datenbanksystem. Diplomarbeit, IKR, University of Stuttgart, 2004.

[69] W3C. Xml schema description language. `http://www.w3.org/XML/Schema`.

[70] Steve Waterhouse, David M. Doolin, Gene Kan, and Yaroslav Faybishenko. Distributed search in peer-to-peer networks. *IEEE Internet Computing*, 1(1):1–5, 2002.

[71] Dave Winer. Xml-rpc specification. `http://www.xmlrpc.com/spec`, June 1999.

Index

A4C, 31, 52, 106
AAA, 31, 33, 52, 107
Accounting, 33
Administrator, 48
AFS, 26, 29
ARP, 24
Autor, 48

Caching, 86
CANDLE, iii, 8, 15, 16, 70, 104, 125, 126, 169
CBT, 10
CIFS, 30
CMS, 19
CODA, 29

DC, 62
DES, 26
DFS, 26, 29
DHT, 38
DLE, 66
DLE-VFS, 84
DublinCore, 20

Edutella, 42

Firewall, 25
Formatvorlagen, 56
FTP, 24

HTTP, 24, 28

ICMP, 24
IETF, 31
IP, 23, 24, 28
ITO, iii, 8, 15, 56, 125, 126, 169
ITU, 31

JXTA, 42

Kerberos, 26

Lehrer, 48
Lehrmodul Transformation, 59, 62
Lehrmodulstrukturen, 55
Lernender, 48
LMS, 12, 16, 52
LOM, 20, 62, 70

Metacoon, 15, 56
Metadaten, 59
MMDB, 15
Mobilität, 49

NFS, 24, 29

Ocean Store, 31

Open Office, 59

P2P, 9, 22, 26, 37, 118

Rahmenbedingungen, 51
RARP, 24
RDF, 21, 36
RFC 1057, 26
RFS, 29
RMI, 27
RPC, 26

SCORM, 20, 62, 70
Semantic Web, 35
SimLib, 109, 151, 152, 157
Smart Cards, 32
SMB, 30
SOAP, 27
SON, 42, 43
SRDI, 40, 110
SSL, 28
STLP, 28
Streams, 28
SWAP, 42

TCP, 23, 24, 28, 111
TFTP, 24
TLS, 28

UDP, 23, 24
Unified Resource Identifier, 70
UPD, 24

VFS, 84, 86
VFSI, 84

W3C, 21
WBT, 10
WEB-DAV, 30
WWW, v, 35

XML, 20, 70
XML-Schema, 70
xmlrpc, 27

Zwischenformat, 66, 70, 73

Anhang A

Simulations- und Messwerkzeuge

Die ereignisbasierte Simulationsumgebung, die für die Bandbreite-Bestimmung des DHT Mechanismus genutzt wurde, basiert auf dem Port-Konzept der IKR eigenen SimLib, ist jedoch eine völlige Neuimplementierung in Java. Dieses Werkzeug bietet die Möglichkeit, die Java-Module des DLE direkt simulativ zu bewerten, bzw. deren Verkehrseigenschaften zu bestimmen. Eine einführende Beschreibung dieser Simulationsbibliothek findet sich in Kapitel A.1.

Desweiteren wurden im Rahmen der Durchführung der vorliegenden Arbeit verschiedene Messmethoden und -werkzeuge evaluiert und implementiert. Hierbei ist zuletzt ein Messwerkzeug entstanden, das es erlaubt, Aufrufe von System- und Bibliotheksfunktionen durchzuführen, ohne hierzu die Applikation selber anpassen zu müssen. Eine Beschreibung dieses Werkzeuges findet sich im Kapitel A.4.

A.1 Event Basierte Simulationsbibliothek

Um einen Eindruck von der Funktionsweise und Nutzung der Simulationsbibliothek zu vermitteln, wird im folgenden ein einfaches Beispiel für die Nutzung betrachtet. Hierbei stehen vor allem die Beschreibung des Aufbaus eines Modells, dessen Konfiguration und die Einflussmöglichkeiten auf die Simulation im Vordergrund.

A.1.1 Komponenten der Simulationsbibliothek

Model Das Simulationsmodell stellt die Basiskomponente der Simulation dar. In ihm werden die Struktur der Simulation und ihre Komponenten definiert und beschrieben. Das Simulationsmodell wird als erstes instanziiert und dann durch die Simulationskontrolle als erste Simulationskomponente in den Lebenszyklus der Simulationskomponenten injiziert. (Siehe auch Abblidung A.9).

Komponenten Sind die Basisobjekte für die Erstellung weiterer Teile einer Simulation.

Ports Die Eigenschaften der Ports oder Schnittstellen der Simulationsbibliothek sind an die Eigenschaften der SimLib des IKR angelehnt. Sie dienen zur Kommunikation von verschiedenen Komponenten der Simulation untereinander. Im besonderen unterstützen sie das „Übertragen von Informationen„ *Messages* und das Anschließen von virtuellen Messgeräten zur Datenaufzeichnung und Überwachung während des Simulationslaufes.

Message- oder Notification-Bus Dieser in die Simulationsbibliothek eingebettete Kommunikationsmechanismus dient zur Zeit maßgeblich dem administrativen Informationsaustausch verschiedener Managementkomponenten des Steuerungswerkes der Simulation. So wird hiermit zum Beispiel die synchronisation der Kompomenten bei Start oder Ende eines Simulationslaufes oder eines Batchs während eines Simulationslaufes gesteuert. Er basiert auf einer minimalen synchronen Implementierung des „Publisher/Subscriber Patterns“, kann aber jederzeit duch einen komplexeren „Messaging“ Provider ausgetauscht werden.

Kalender Der Kalender ist die zentrale Komponenete der Zeiterfassung, sowie der zentrale Steuermechanismus für die zeit-richtige Abarbeitung der im System entstehenden Ereignisse. Die gewählte Implementierung ist an den Vorschlag eines „Fast Calendars“ aus [7] angelehnt. In Erweiterung der Kalender-Events aus der SimLib unterstützt die hier gewählte Implementierung die Mitgabe von Call-Back-Funtionen und deren Parametrisierung.

Simulationskontrolle Die Simulationskontrolle dient der Steuerung der Phasen der Simulation. Dies gilt für die Initialisierungs- , Konfigurations- und Startphase der Simulation, in der auch die Komponenten in den jeweiligen Zustand gebracht werden. Darüber hinaus steuert sie die Statistiken im Rahmen der Transienten-Phase sowie der Batch-Phasen und verteilt die dazu notwendigen Signale. Dies gilt im besonderen auch für die Erzeugung von Ergebnis- und Logfiles.

Konfiguration Der Konfigurationsmechanismus dient zum Setzten von Simulationsparametern der in der Simulation verwendeten Komponenten. Sie ist in XML definiert und hierarchisch, wie das Simulationsmodell, organisiert. Der gewählte Konfigurationsmechanismus erlaubt das Setzen beliebiger Attribute auch auf erweiterten Komponenten der Implementierung sowie das Verwalten von Laufparametern der Simulation und frei erweiterbarer Variablenauflösung und Ausdruckverarbeitung.

A.1.2 Die Konfigurationsdatei

Das Konfigurationsfile dient der Parametrisierung der Simulationskomponenten sowie der Steuerung abhängiger Simulationsläufe. Hierzu können im Konfigurationsfile Laufvariablen und deren Werteabfolge definiert werden, welche dann, von der externen Simulationskontrolle in der definierten Abfolge gesteuert, durchlaufen werden. Nach Ablauf aller Simulationen kann auch die Auswertung der erzeugten Simulationsergebnisse aus dem Konfigurationsfile heraus gesteuert werden.

Die Konfigurationsdatei besteht hierbei aus den im folgenden Abschnitt beschriebenen Blöcken.

Konfigurationsstruktur

Das Konfigurationsfile teilt sich in die Blöcke

- Simulationssteuerung
- Globale Variablen und Lauf Variablen

- Parametrisierung der Simulationskomponenten

- Ergebnis Verarbeitung

Für eine vollständige Übersicht über die Struktur und Syntax der Konfigurationsdatei ist deren Definition in den Abblidungen D.5 und D.6 angegeben.

Hierbei werden bei der Verarbeitung der Konfigurations-Datei die in den folgenden zwei Abschnitten dargestellten Mechamismen zur Variablen-Auflösung und der Komponenten-Parametrisierung genutzt.

Variablenauflösung und Ersetzen von Ausdrücken

Die Parameter-Werte lassen sich innerhalb des Konfigurationsfiles nicht nur als direkte Werte angeben, sondern es existiert die Möglichkeit, auf bereits definierte Variablen und hinterlegte Funktionen zur deren Berechnung zurückzugreifen. Die Syntax zum Aufruf des Variablen- und Ausdruck-Ersetzers sind in den eigentlichen Wert eingesetzte Steuersequenzen der Form ${}. Der dazu genutzte Funktions und Variablen-Interpreter ist hierbei beliebig erweiterbar gehalten.

Konfigurationsmechanismen

Auf Basis der oben beschriebenen Zuordnungen von Werten zu Parametern in der Simula-tions-komponeten-Hierarchie werden dann in der nachfolgend beschriebenen Konfigurationsphase der Simulation mit dem in Abbildung A.1 dargestellten Mechanismus die Werte der Parameter der Simulationskomponete gesetzt. Die Konfigurationsphase ist eine der Phasen des Lebenszykluses einer Komponente. Die einzelnen Phasen sind hierbei im nächsten Abschnitt beschrieben.

```
private boolean setVariable(String name, String value) {
      // Used by Configurable.configure(Configuration cfg)
      // Find a matching setter ...
      if (LOG.isLoggable(Level.FINEST))
         LOG.finest("Setting Variable " + name + " in " + this.getName());
      boolean ret = false;
      Class tc = this.getClass();
      Method[] methods = tc.getMethods();
      String mname = "set" + name.toUpperCase().substring(0, 1)
            + name.substring(1);
      for (Method m : methods) {
         if (mname.equals(m.getName())) {
            Class[] pts = m.getParameterTypes();
            if (pts.length == 1) {
               Class pt = pts[0];
               Object o = Converters.convert(value, pt);
               try {
                  m.invoke(this, new Object[] { o });
                  ret = true;
                  break;
               } catch (Exception e) {
                  e.printStackTrace();
               }
            }
         }
      }
      if (!ret) {
         try {
            Field f = tc.getDeclaredField(name);
            f.setAccessible(true);
            Class fc = f.getType();
            Object o = Converters.convert(value, fc);
            f.set(this, o);
            ret = true;
         } catch (Exception e) {
            if (LOG.isLoggable(Level.FINER))
               LOG.throwing(this.getClass().getName(), "Could not Set: "
                     + name, e);
         }

      } return ret;   }
```

Abbildung A.1: Die Komponenten-Konfiguration

A.1.3 Lebensphasen der Komponenten

Die folgenden Abschnitte beschreiben in Kürze die Phasen,die jede Simualationskomponente während ihres „Lebens“ durchlaufen muss, sowie die Aufgaben, die während der jeweiligen Phase erledigt werden sollten.

Initialization In der Initialisierungsphase sollten alle während der Lebensdauer der Komponente notwendigen Resourcen bereitgestellt werden.

Configuration Wärend der Konfigurationsphase werden alle existierenden Resourcen mittels der Informationen aus dem Konfigurationsfile parametrisiert.

Start In der Startphase sollten danach alle für den Ablauf der Simulation notwendigen Anfangsereignisse in den Kalender eingestellt werden. Ist die Startphase abgeschlossen, wird die Abarbeitung der Kalenderereignisse, respektive der Kalender der Simulation, gestartet.

Pause In einer „Pause“-Phase wird der Simulationslauf unterbrochen. Hier sollten alle von den Kalenderereignissen unabhängigen Teilprozesse ebenfalls angehalten werden. Aus der Pausen-Phase wird die Simulation mit einem erneuten Start-Aufruf wieder gestartet.

StartBatch Die StartBatch-Phase dient nach Abschluss einer inhärenten transienten Phase oder einer StopBatch-Phase zur entsprechenden Initialisierung der Statistiken der Simulation.

StopBatch Während einer StopBatch-Phase werden die noch zum jeweilig laufenden Batch der Simuation gehörigen Events abgearbeitet, die Statistiken für den Batch berechnet und die Ergebnisse des Batches zur Ausgabe bereitgestellt.

Stop Die Stop-Pase dient dazu allen Komponenten die Möglichkeit zu geben, ihren Lebenszyklus zu beenden und eventuell Resourcen wieder freizugeben. Darüber hinaus werden die Statistiken der Simulation finalisiert und die Gesamtergebnisse für die Ausgabe bereitgestellt.

A.2 Beispiel eines Modells

Das im Folgenden beschriebene Modell dient zur Verdeutlichung der Nutzung der Java-SimLib. Es werden an einem sehr simplen Beispiel die Mechanismen zur Erstellung, Konfiguration und Ablaufsteuerung einer Simulation beschrieben.

Beginnend mit der Erstellung des Basis-Simulationsmodells im anschließenden Unterkapitel, über die Erstellung der Konfigurations-Datei und dem Startscript, wird im letzten Teilkapitel noch auf die Einflussmöglichkeiten auf die Simulation währed deren Laufzeit eingegengangen.

A.2.1 Aufbau des Modells

Das Modell welches hier aufgebaut werden soll, ist in Abbildung A.2 dargestellt.

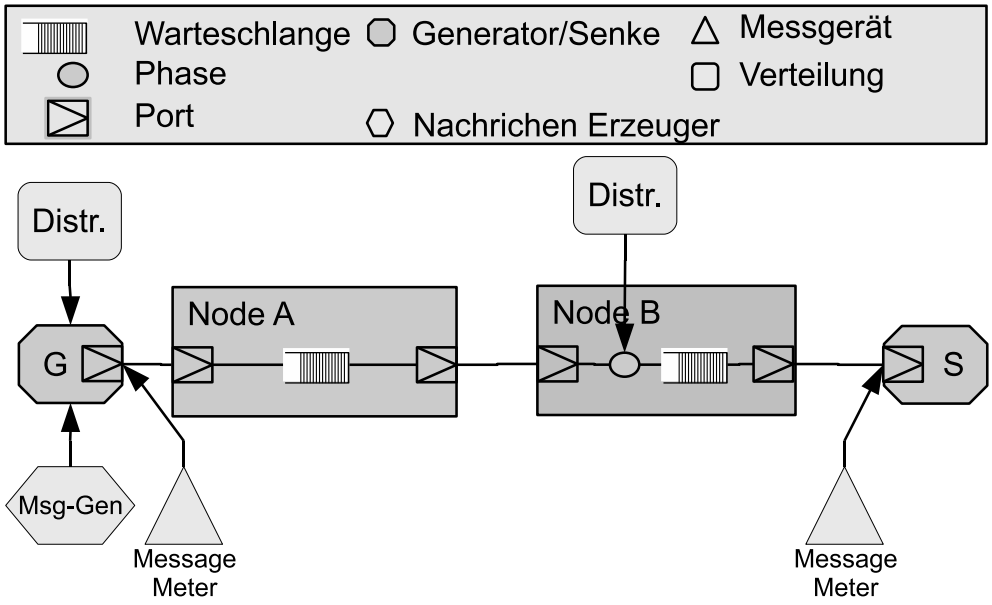


Abbildung A.2: Modell der Beispielsimulation

A.2.2 Implementierung des Modells

Um das dargestellte Modell in eine Simulation umzuwandeln, müssen die Klassen für das Simulationsmodell sowie die zwei Knoten geschrieben werden. Darin werden dann die weiteren Komponenten, Distributionen und Messgeräte instanziiert und miteinander verbunden. Die Parametrisierung der Komponenten erfolgt dann über das Konfigurationsfile.

Das Modell der Simulation wird als Ableitung der Klasse *SimModel* A.2 wie in A.5 dargestellt geschrieben.

Die Implementierung der zwei beteiligten Knoten wird dann analog als Ableitung der Klasse *SimEntity* (wie in Abbildung A.6 dargestellt) implementiert.

A.2.3 Konfiguration des Simulationsmodelles

Die Konfiguration der Simulation erfolgt dann über die in Abbildung A.7 dargestellte Konfigurationsdatei.

A.3 Einflussmöglichkeiten auf die Simulation

Die zur Implementierung der vorgestellten Simulationsbibliothek genutzte Programmiersprache Java bietet über die bereits beschriebenen Möglichkeiten den Vorzug, dass die Simulation sehr leicht an mannigfaltige Anwendungsanforderungen angepasst werden kann. Es ergeben sich Möglichkeiten der Beeinflussung der Simulation durch grafische Komponenten, die „real-time“ Anzeige der Statistiken oder bestimmter Messwerte der Simulation oder auch die Nutzung der Bibliothek für interaktive Visualisierungen komplexer Systeme oder Zusammenhänge.

Ein Bespiel für eine solche Anwendung ist zum Beispiel die in Abbildung A.10 dargestellte Demonstrations-Anwendung zur Visualisierung von Linearen Schieberegistern wie sie zum Beispiel in der Vorlesung „Error Detecting und Error Correcting Codes“ eingesetzt

```
/*
 * Created on 14.07.2004
 *
 *
 */
package fi.promo.sim.model;

import fi.promo.sim.component.Initiable;
import fi.promo.sim.component.SimComponent;
import fi.promo.sim.control.SimulationControl;


/**
 * @author Lutz Finsterle
 *
 *
 */
public abstract class SimModel extends SimComponent implements Initiable {

        /**
         *
         */
        public SimModel(String name) {
                super(name, null);
                SimulationControl.registerModel(this);
        }

}
```

Abbildung A.3: Basisklasse SimModel

werden könnte.

A.4 Messwerkzeug für Prototypen

Im Rahmen der vorliegenden Arbeit wurden verschiedene Messungen an dem realisierten Prototypen durchgeführt. Um die Messwerterfassung und -auswertung möglichst einfach, jedoch sehr flexibel, zu gestalten, wurden hierfür ein Messwerkzeug entwickelt. Dieses erlaubt es auf einfache Art und Weise die benötigten Messungen durchzuführen

und die relevanten Messwerte zu erfassen.

Bei der Realisierung dieses Werkzeuges wurde im besonderen darauf Wert gelegt, dass zu beobachtende Programm so wenig wie möglich zu beeinflussen. Um trotz dieser Anforderung eine breite Palette an Daten erfassen zu können, wurde die Technik des Abfangens von C-Bibliotheksaufrufen angewandt. Um die Messdaten dabei möglichst flexibel aus dem Messwekzug abgreifen zu können, wurde für die Kommunikation der Messbibliothek zum Erfassungswerkzeug die Kommunikation über IPC-Messages gewählt. Dieses hat vor allem auch den Vorzug, dass alle anderen, hauptsächlich genutzten Kommunikationspfade, unbeeinflusst bleiben. Eine grobe Übersicht über die Architektur ist in Abbildung 6.11 dargestellt.

Die folgende Abbildung zeigt auf, wie sich die Messbibliothek erweitern lässt. Es muss hierzu, analog zu der Methode „send“, für die gewünschte Methode ein entsprechder Wrapper geschrieben werden. Ebenso einfach ist das Anpassen der Log-Nachrichten an ein anderes Format.

```
1  package fi.sim.test;
2
3  import ...;
4
5  public class QueuePhaseTest extends SimModel {
6
7     private boolean initialized;
8     private StdGenerator testGen;
9     private Sink testSink;
10    private Distribution dist;
11    private TrafficMeter meter1;
12    private TrafficMeter meter2;
13    private TestBasicMessageMeter mf1;
14    private TestBasicMessageMeter mf2;
15    private SimControlTimer sicc;
16    private TestNodeA nodeA;
17    private TestNodeB nodeB;
18
19    public QueuePhaseTest(String name) {
20       super(name);
21       meter1 = new TrafficMeter("GenMeter");
22       meter2 = new TrafficMeter("SinkMeter");
23       System.err.println("Create Node1");
24        nodeA = SimEntityFactory.create(TestNodeA.class, "TestNodeA",
25             this);
26        nodeB = SimEntityFactory.create(TestNodeB.class, "TestNodeB",
27             this);
28       testGen = SimEntityFactory.create(StdGenerator.class, "TestGen", this);
29       dist = new NegExpDistribution("GenDist",1000);
30       testGen.setDistribution(dist);
31       testGen.setMessageCreator(new TestMsgCreator());
32       testSink = SimEntityFactory.create(Sink.class, "TestSink", this);
33       try {
34          testGen.portName2Port("output").addMessageFilter(
35                mf1 = new TestBasicMessageMeter(this, "mfGen"));
36          testSink.portName2Port("input").addMessageFilter(
37                mf2 = new TestBasicMessageMeter(this, "mfSink"));
38       } catch (PortException e1) {
39          e1.printStackTrace();
40       }
41    }
```

Abbildung A.4: MySimModel

```
46     public void pause() {
47     }
48
49     public void reset() {
50     }
51
52     public void start() {
53        System.err.println("Starting sim: " + getName());
54        try {
55           connect(testGen, "output", nodeA, "input");
56           connect(nodeA, "output", nodeB, "input");
57           connect(nodeB, "output", testSink, "input");
58        } catch (PortException e) {
59           e.printStackTrace();
60        }
61     }
62
63     public void stop() {
64     }
65
66     public void init(ParameterMap p) {
67        if (!isInitialized()) {
68           System.err.println("init");
69           sicc = new SimControlTimer(1000000, 1000, 100000);
70           initialized = true;
71        }
72     }
73
74     public boolean isInitialized() {
75        return initialized;
76     }
77  }
```

Abbildung A.5: MySimModel cont.

```
1  package fi.sim.test;
2
3  public class TestNodeA extends SimComponent implements Initiable, Configurable,
4        Startable {
5
6     private InputPort inPort;
7     private OutputPort outPort;
8     private UnboundedFifoQueue testQueue;
9     protected boolean initialized;
10
11    public TestNodeA(String name, SimEntity owner) {
12       super(name, owner);
13       testQueue = SimEntityFactory.create(UnboundedFifoQueue.class, "AQueue",
14             this);
15    }
16
17    public TestNodeA(String name) {
18       this(name, null);
19    }
20
21    public void start() {
22
23    }
24
25    public void stop() {
26
27    }
28
29    public void init(ParameterMap p) {
30       if (!isInitialized()) {
31          aliasPort(testQueue, "input", "input");
32          aliasPort(testQueue, "output", "output");
33          initialized = true;
34       }
35    }
36
37    public boolean isInitialized() {
38       return initialized;
39    }
40
41 }
```

Abbildung A.6: Ein einfacher Knoten für das Simulationsmodell

```
<?xml version="1.0" ?>
<!DOCTYPE Simulation SYSTEM "C:\Development\Diss33\FISimTests\configure\simConfig.dtd">
<Simulation>
  <GeneralInformation>
    <Title>Simulation Test</Title>
    <SimulationTool>fi.sim.test.QueuePhaseTest</SimulationTool>
    <ResultsDirectory>
      /tmp/SimTest2/Results_20070829
    </ResultsDirectory>
  </GeneralInformation>

  <ParameterSet>
    <FixedParameter Name="TransientEvents">10000</FixedParameter>
    <FixedParameter Name="NumberOfBatches">10</FixedParameter>
    <FixedParameter Name="EventsPerBatch">1000000</FixedParameter>
    <FixedParameter Name="TransientTime">100000</FixedParameter>
    <FixedParameter Name="TimePerBatch">1000000</FixedParameter>
    <FixedParameter Name="startupTime">20000</FixedParameter>
    <FixedParameter Name="numReplicas">4</FixedParameter>
  </ParameterSet>

  <SimulationModel Name="QueuePhaseTest">
    <ParameterSet></ParameterSet>
    <Entity Name="TestNodeA">
      <ParameterSet></ParameterSet>
      <Entity Name="QueueA">
        <ParameterSet></ParameterSet>
      </Entity>
    </Entity>

    <Entity Name="TestNodeB">
      <ParameterSet></ParameterSet>
      <Entity Name="TestPhase">
        <ParameterSet>
          <FixedParameter Name="unitDelay">100</FixedParameter>
        </ParameterSet>
        <Distribution Name="PDist">
          <ParameterSet>
            <FixedParameter Name="mean">40</FixedParameter>
          </ParameterSet>
        </Distribution>
      </Entity>
    </Entity>

```

Abbildung A.7: Konfiguration MySimModel

```
45      <Entity Name="mfGen">
46        <ParameterSet>
47          <FixedParameter Name="logData">false</FixedParameter>
48        </ParameterSet>
49
50
51        <Entity Name="RateMeter">
52          <ParameterSet>
53            <FixedParameter Name="showGraph">
54              true
55            </FixedParameter>
56          </ParameterSet>
57        </Entity>
58      </Entity>
59
60      <Entity Name="mfSink">
61        <ParameterSet>
62          <FixedParameter Name="logData">false</FixedParameter>
63        </ParameterSet>
64
65        <Entity Name="RateMeter">
66          <ParameterSet>
67            <FixedParameter Name="showGraph">
68              false
69            </FixedParameter>
70          </ParameterSet>
71        </Entity>
72      </Entity>
73
74    </SimulationModel>
75    <Evaluation></Evaluation>
76
77  </Simulation>
```

Abbildung A.8: Konfiguration MySimModel

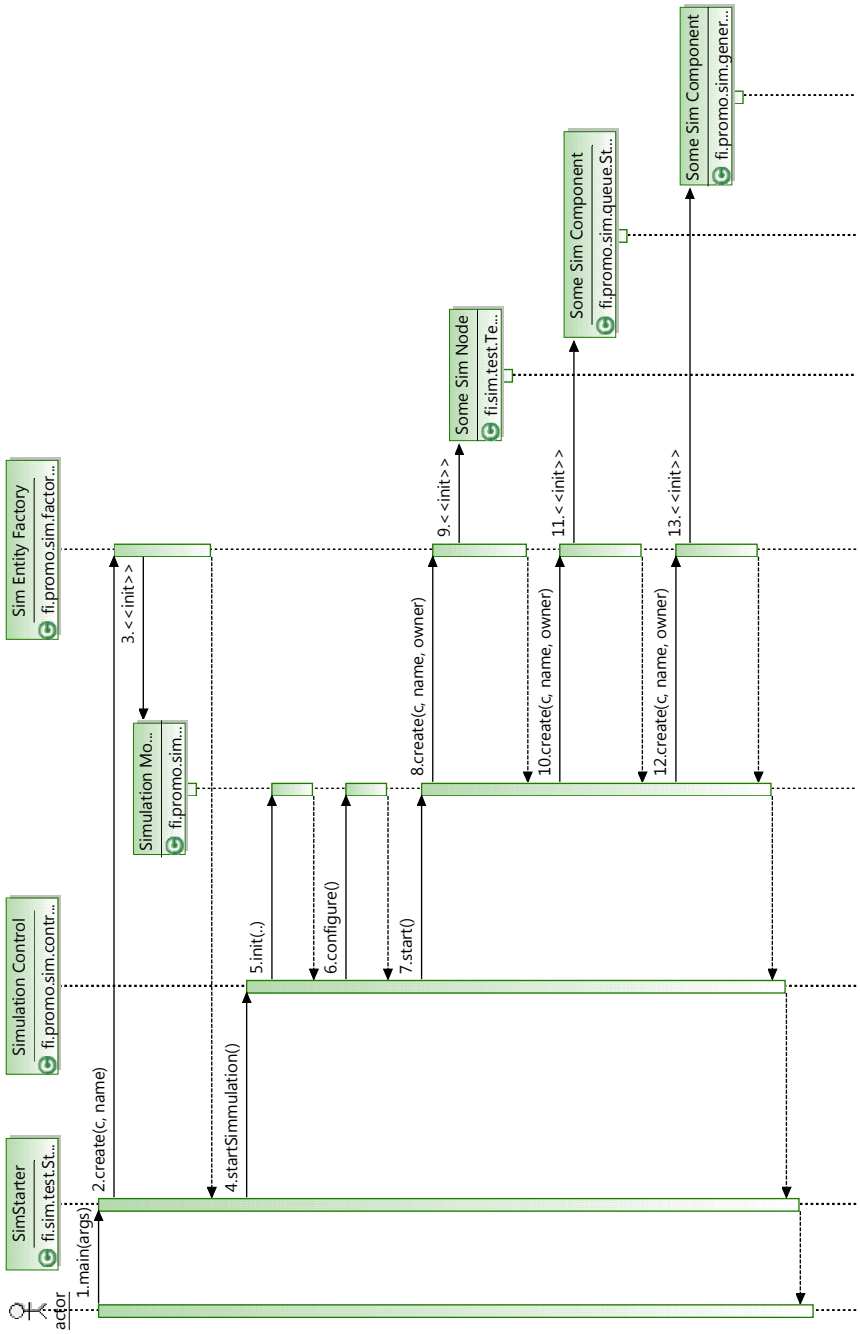


Abbildung A.9: Startup Simulation Sequence Diagramm

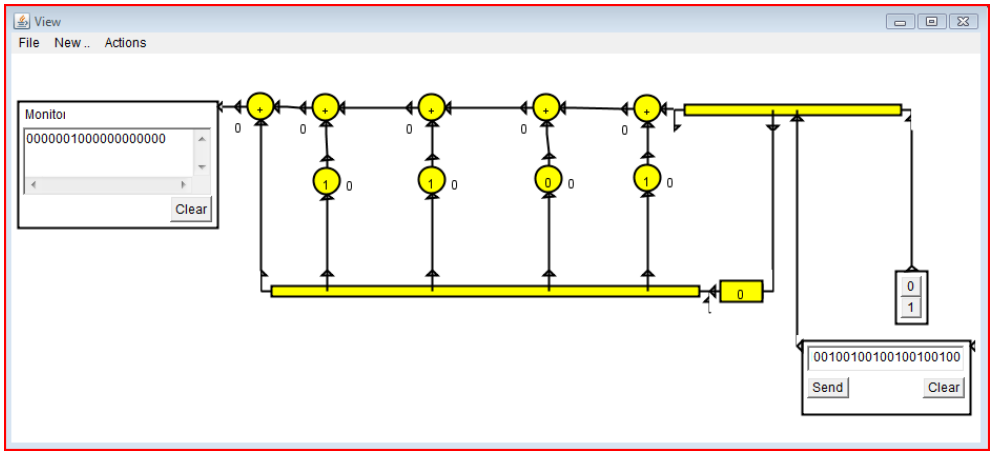


Abbildung A.10: Interaktive Lehranwendung

```
ssize_t send(int fd, const char* buffer, size_t count, int flags){
    static ssize_t (* func)(int,const char* b, size_t, int flags)  = 0;

 if (func == 0)
    func = (ssize_t (*) (int,const char *,size_t, int))
                                    dlsym( RTLD_NEXT , "send");



 int ret = (*func)(fd,buffer,count,flags);
 if (isRegisteredFileDescriptor(fd)){
    char _add[100];
    snprintf(_add,99,"Send ( %d from %d) bytes to %d",ret,count,fd);
    writeLog(fd,"cafdboioad","send",fd,ret,ret,_add);
 }
 return ret;
}
```

Abbildung A.11: Abfangen der send() Methode

Anhang B

Kurse und Kursstrukturen

In diesem Teil des Anhangs werden kurz die wichtigsten Eigenschaften der in den Projekten CANDLE und ITO sowie am Lehrstuhl für Kommunikationsnetze und Rechnersysteme im Rahmen der E-Learning Evaluation untersuchten und aufbereiteten Vorlesungsmaterialien dargelegt. Die Kurse wurden hierbei von verschiedenen Universitäten im Rahmen der oben genannten Projekte erstellt bzw. an die Anforderungen des jeweiligen Projektes angepasst. Die Kurse und ihre jeweiligen Autoren sind in Tabelle B.1 aufgeführt.

Für die vorliegende Arbeit sind maßgeblich die Größe und die Anzahl der Module der modularisierten Vorlesungsmaterialien von Belang, da diese Informationen einen wesentlichen Einflussfaktor für die Netzlastabschätzungen darstellen. Hierbei muss zwischen den Größeninformationen der Metadaten, der Textdaten und der Multimedia-Anteile unterschieden werden, da diese, aufgrund der verschiedenen Verwendungen, unterschiedlichen Einfluss auf die Netztlast des Gesamtsystems haben. Metadaten beispielsweise, werden für alle Such- und Strukturanfragen benötigt, wohingegen die textuellen Daten der Vorlesungen im Rahmen von „Suche“, wenn überhaupt, dann nur für spezielle Volltextsuchanfragen benötigt werden. Damit ist die Notwendigkeit des Zugriffes auf die Text- und Multimedia-Anteile nur für die Fälle Authoring und Viewing notwendig, welche sich leicht durch die beschriebenen Caching-Mechanismen (Siehe Kapitel 5.4.5) reduzieren lassen. Die Metadaten der Vorlesungsmodule müssen jedoch häufiger

Tabelle B.1: Kurse und deren Autoren

Kurs	Autoren	Universität	Land
Kurse aus CANDLE:			
Error Detecting and Error Correcting Codes	Paul J. Kühn	Universiät Stuttgart	Deutschland
Kommunikation und Datenhaltung	Sebastian Abek	Universität Karlsruhe	Deutschland
Communication Networks I	Paul J. Kühn	Universiät Stuttgart	Deutschland
Communication Networks II	Paul J. Kühn	Universiät Stuttgart	Deutschland
Elements of Queing Theory	Rolv Break	NTNU	Norwegen
Engineering Distributed Real Time Systems	Rolv Break	NTNU	Norwegen
Telematics Applications	Sebastian Abeck	Univerität Karlsruhe	Deutschland
Lecture Managed IT Systems	Sebastian Abeck	Univerität Karlsruhe	Deutschland
Mobile Communications	Lluis Gutierez	Universita Politecnica de Catalunja	Spain
Network Management	Lluis Gutierez	Universita Politecnica de Catalunja	Spain
Telcomunication Systems (coarse grain)	Rolv Break	NTNU	Norwegen
Telcomunication Systems (fine grain)	Rolv Break	NTNU	Norwegen
Kurse aus ITO:			
Error Detecting and Error Correcting Codes	Paul J. Kühn	Universität Stuttgart	Deutschland
Communication Networks I	Paul J. Kühn	Universität Stuttgart	Deutschland
Communication Networks II	Paul J. Kühn	Universität Stuttgart	Deutschland
Visualisation Systems	Ertl	Universität Stuttgart	Deutschland

geprüft werden, da sie auch für die Verwaltung, wie zum Beispel die gerade erwähnten Cachingmechanismen, wichtig sind. In Tabelle B.2 sind die entsprechenden Kennzahlen für die Objektgrößen aufgeführt.

Tabelle B.2: Kurs- und Metadaten-Größen

Kurs	**Anzahl der Metadaten Einträge**	**Größe der Inhalts-Einträge (Summe [Byte])**	**Größe der Metadaten-Eintrage (Summe [Byte])**
Kurse aus CANDLE:			
Error Detecting and Error Correcting Codes	91	229579	2522
Kommunikation und Datenhaltung	37	743339	6748
Communication Networks I	43	7655453	97886
Communication Networks II	62	12346098	112398
Elements of Queueing Theory	39	1249332	30129
Engineering Distributed Real Time Systems	51	8675243	1678
Lecture Managed IT Systems	30	554456	1848
Mobile Communications	146	122553	839
Network Management	49	9775422	147986
TelSys (coarse grain)	12	29285	2440
Tel Sys (fine grain)	53	61130	1153
Kurse aus ITO:			
Error Detecting and Error Correcting Codes	91	229579	2522
Communication Networks I	43	7655453	97886
Communication Networks II	62	12346098	112398
Visualisation Systems (Ertl)	104	19447932	139122

Die im Tabelle B.2 sind hierbei im den Projekten CANDLE und ITO hierbei wie in den folgenden Beispielen aufgezeit erstellt worden. Die Beispiele geben hierbei einen Einblick in die Verwendeten Werkzeuge sowie in die Struktur der Aufbereiteten Lerninhalte an sich. Es wird die Aufbereitung der Inhalte dabei nur soweit exemplarisch wiedergegeben wie es zum Verständnis der oben dargelegten Kennzahlen notwendig ist.

B.1 Aufbereitung der Lernmaterialien in CANDLE

Die Aufbereitung der Lernmaterialien in CANDLE wurde mit dem im Projekt erstellten Werkzeug CAT (Candle Authoring Tool) durchgeführt. Hierzu ist es nötig, die vorliegenden MAterialien in ihre Bestandteile zu zerlegen und dann mit Hilfe von CAT zu Annotieren und zu Strukturieren. In den Abbildugen B.1 B.2 B.3 und B.4 sind hierzu die ur-

sprügliche Form, die aufgespaltene Form, ein Prinzipauszug aus den Metadaten sowie eine Darstellung der CAT-Werkzeugens dargestellt.

B.2 Aufbereitung der Lernmaterialien in ITO

Das Vorgehen für die Materialen innerhalb des ITO Projektes basiert auf einer annähernd gleichen Vorgehensweise wie im vorangegangenen Abschnitt für das Projekt CANDLE dargestellt wurde. Einzige Unterscheidung ist das verwendendete Autorenwerkzeug das für den Fall ITO bereits im Kapitel 4 eingehend vorgestellt wurde.

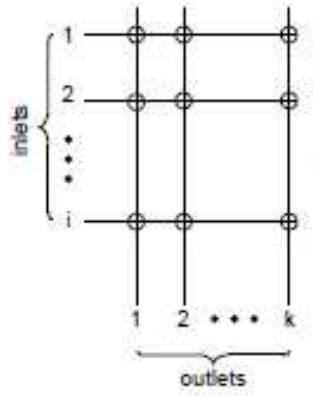
B.3 Berechnung der Inhalts- und Metadatengrößen

Die Berechnung der Metadaten und Lehrmodulgrößen wurde über die in den Repositories befindlichen Datensätze, respektive über die Filegrößen in den lokalen Repositories durchgeführt. Die Größenberechnung der Originaldokumente wurde anhand deren Größe im jeweiligen Quellformat im Filesystem durchgeführt.

1.1 Single-Stage Switching Arrays

1.1.1 Crossbar

The elementary single-stage switching array is a matrix of crosspoints, also called "crossbar":



i: number of incoming circuits (lines, trunks, inlets)

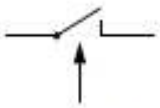
k: number of outgoing circuits (lines, trunks, outlets) also called : accessibility.

Crosspoint

A circuit-switched connection between an incoming circuit and an outgoing circuit is realized by closing the crosspoint switch located at the intersection of the incoming and outgoing circuit.

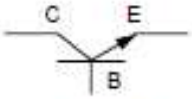
Realization of Crosspoints:

Metallic Contact



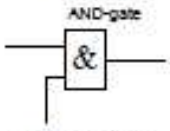
Control by a Relay

Electronic Contact (analog)



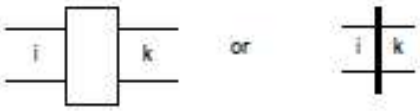
Control by Base Current of a Transistor

Electronic Contact (digital)



Control by logic value "1"

Symbol of single-stage matrix (crossbar)



k is called "accessibility"

The crossbar has "full accessibility", as any outgoing circuit can be reached from any incoming circuit if idle

Abbildung B.1: Ursprüngliches Vorlesungsmaterial

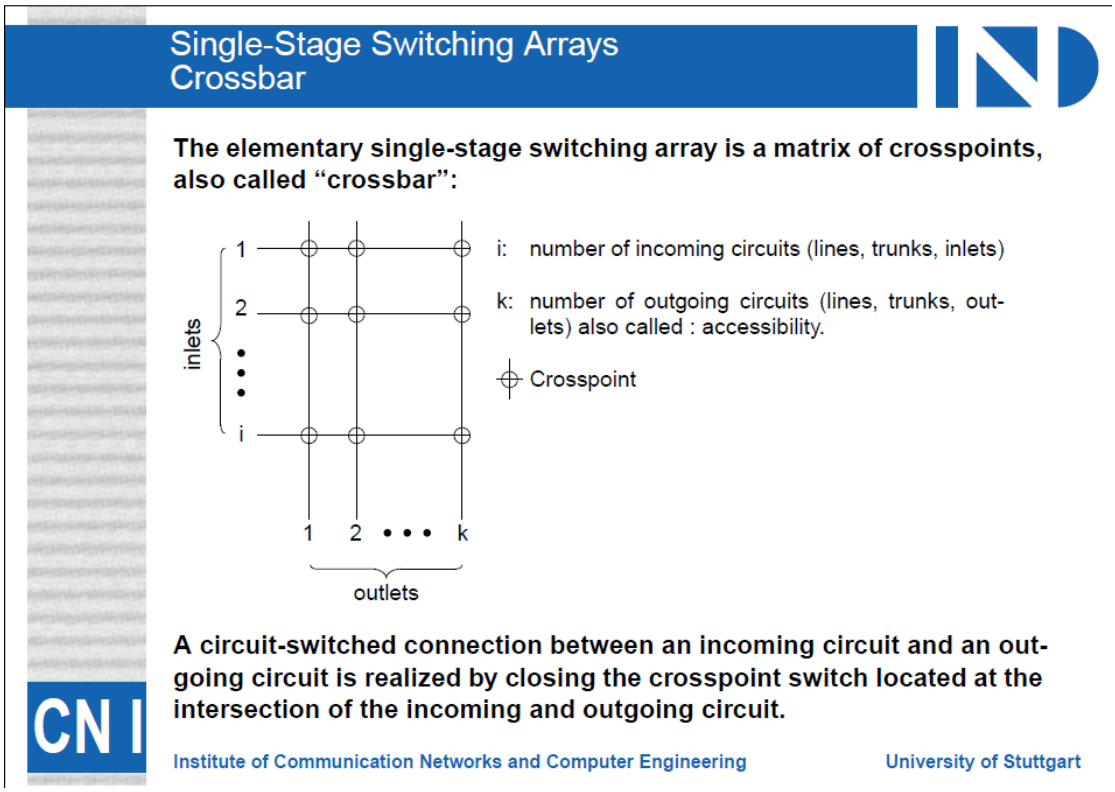


Abbildung B.2: Aufbereitetes Vorlesungsmaterial

```
<?xml version="1.0" encoding="UTF-8"?>
<meta>
    <general>
        <title>Crossbar</title>
        <author>
            <firstname>Paul J.</firstname>
            <lastname >Kuehn</lastname >
            <institute>IKR</institute>
            <contact>email</contact>
        </author>
        <description />
        <language>en-us</language>
        <granularity>C-Module</granularity>
    </general>
    <lifecycle>
        <date>11/20/2002</date>
        <version >3.1</version>
    </lifecycle>
    <pedagogical>
        <structure>linear</structure>
        <context>
            <mode>presentation</mode>
            <whoTo>Students</whoTo>
        </context>
    .
    .
    .
    </pedagogical>
    <technical>
        <location>1.ukaib01-1039953631531-2595</location>
        <format />
    </technical>
    <classification>
        <keyword />
    </classification>
    <metameta>
        <uid>2.ukaib01-1039953635390-1756</uid>
    </metameta>
</meta>
```

Abbildung B.3: Exemplarischer Metadatenauszug

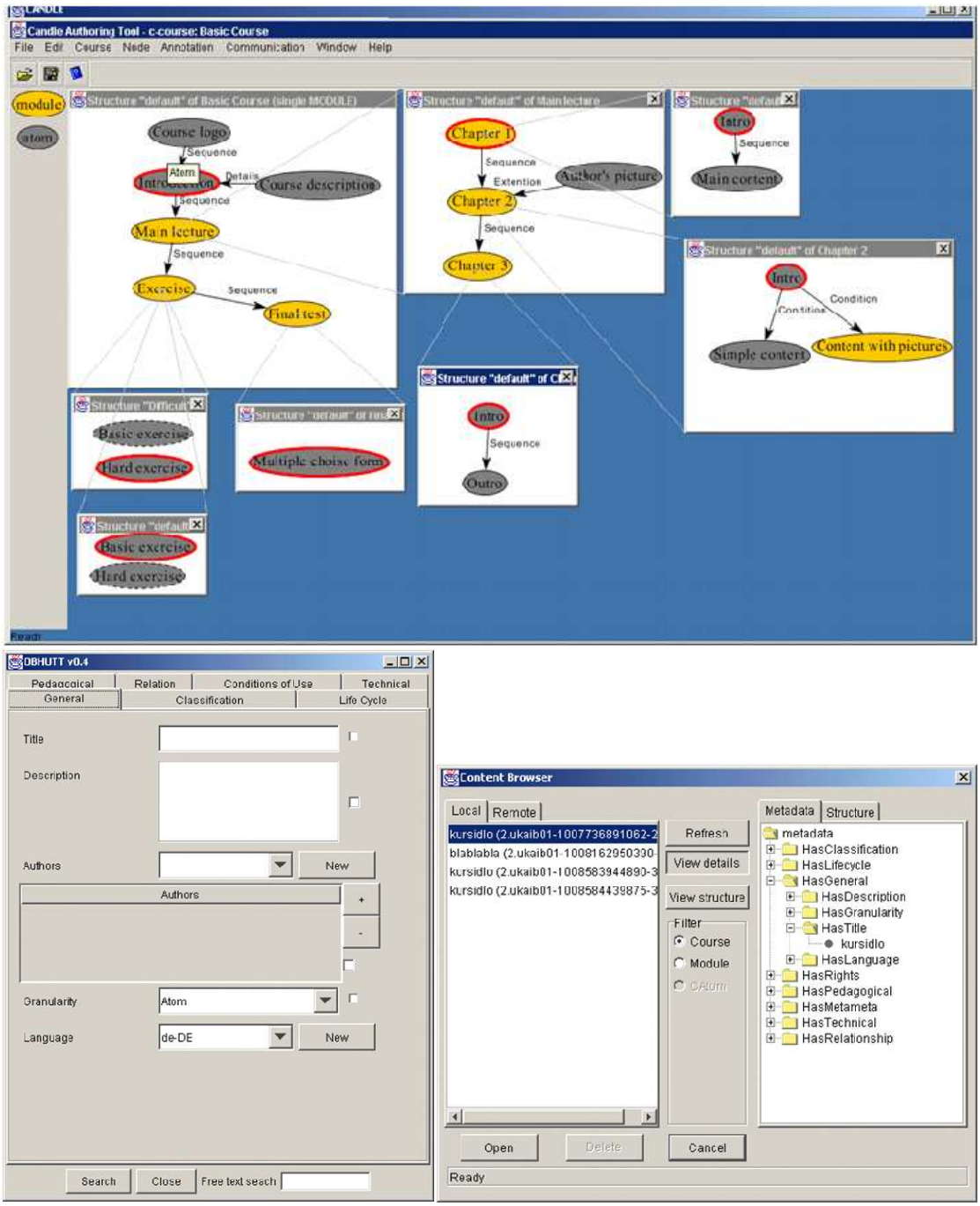


Abbildung B.4: Das Candle Authoring Tool (CAT)

Anhang C

Anwendungen und deren Anforderungen

Die Palette der E-Learning Applikationen ist sehr groß. Allerdings lassen sich diese Anwendungen in wenige Kategorien einteilen. Die Kategorien beherbergen dann solche Applikationen, welche nahezu gleiche Anforderungen an zum Beispiel das Kommunikationssystem, das Personalisierungsystem, die Datenverteilungs- oder die Sicherheitsmechnismen stellen.

Die Anwendungsklassen und deren Anforderungsprofil, in welche die Anwendungen eingeteilt werden sind:

A: Editoren oder Autorensysteme: In diese Kategorie fallen alle an das System angebundenen Werkzeuge zur Inhaltserstellung und Inhaltsannotation. Diese stellen besonders hohe Anforderungen an die Content Management Ebene sowie an die Zugriffsschicht auf Personalisierung und Profildaten.

B: Simulationen und Virtuelle Experimente: Diese Kategorie beschreibt interaktive Applikationen, welche auf dem Rechner des Dienstnutzers laufen, dabei aber ständig mit dem Dienstanbietenden Server in Verbindung stehen. Der Datenaustausch muss hierbei so schnell stattfinden, dass die Anwendung für den Nutzer gut bedienbar ist. Je nach Ausprägung der Anwendung kann dies eine Echtzeitfähigkeit des Informationsaustau-

sches bedeuten.

C: Portale und Präsentationsanwendungen: Portale und Präsentationsanwendungen stellen im Bereich der Personalisierung und der Aufbereitung der Materialien hohe Ansprüche.

D: Chat-, Konferenz- oder Groupware-Anwendugen: Diese Gruppe benötigt im Besonderen die Möglichkeit des Multicasts und des Software-Zugriffes.

E: Audio- oder Videoströme: Diese Klasse an Anwendungen, welche zum Beispiel für die Nachbereitung von Vorlesungen, unterstützt durch die Aufzeichnung derselben, wichtig ist, stellt die höchsten Anforderungen an die Bandbreite und die Echtzeitfähigkeit der Transportschicht.

Zusammengefasst kann also die folgende Aufstellung an Anforderungen and die DLE-System-Komponenten extrahiert werden. In Tabelle C.1 sind die Anforderungen zusammengestellt.

Tabelle C.1: Anforderungen der Anwendungen

Systemische Anforderung	**Ursprung**
Filesystem Zugriff auf das Inhaltsrepository	AC
Filesystem Zugriff auf Profildaten	ABC
Zugriff auf Anwendungserweiterungen	ABC
HTML Rendering	C
Softwarezugriff	B
Multicast	D
Echtzeit Datenstreaming	E,D

Anhang D

Ergänzungen

D.1 Formatdefinition des Zwischenformates

In Ergänzung zu den in Kapitel 4.2 dargelegten Ausführungen sind hier die Definitions-Dateien für das beschriebene Zwischenformat zur Speicherung der Lehrinhalte im DLE wiedergegeben. Die hier abgedruckten Dateien zeigen den angestrebten modularen Aufbau des Zwischenformates und die möglichen Erweiterungspunkte auf. Weiterhin wird an den Strukturen deutlich, wie sich die notwendigen Metadaten in das Konzept integrieren und wie die Trennung zwischen Form und Inhalt erreicht wird.

D.2 Vollständige Übersicht über die Basis-Datenstrukturen des DLE

Die folgenden Bilder geben die Komplexität der Basisdatenstruktur des DLE's wieder. Sie sind die grafische Darstellung der in Kapitel 5.5 beschriebenen Datenstruktur-Konfiguration.

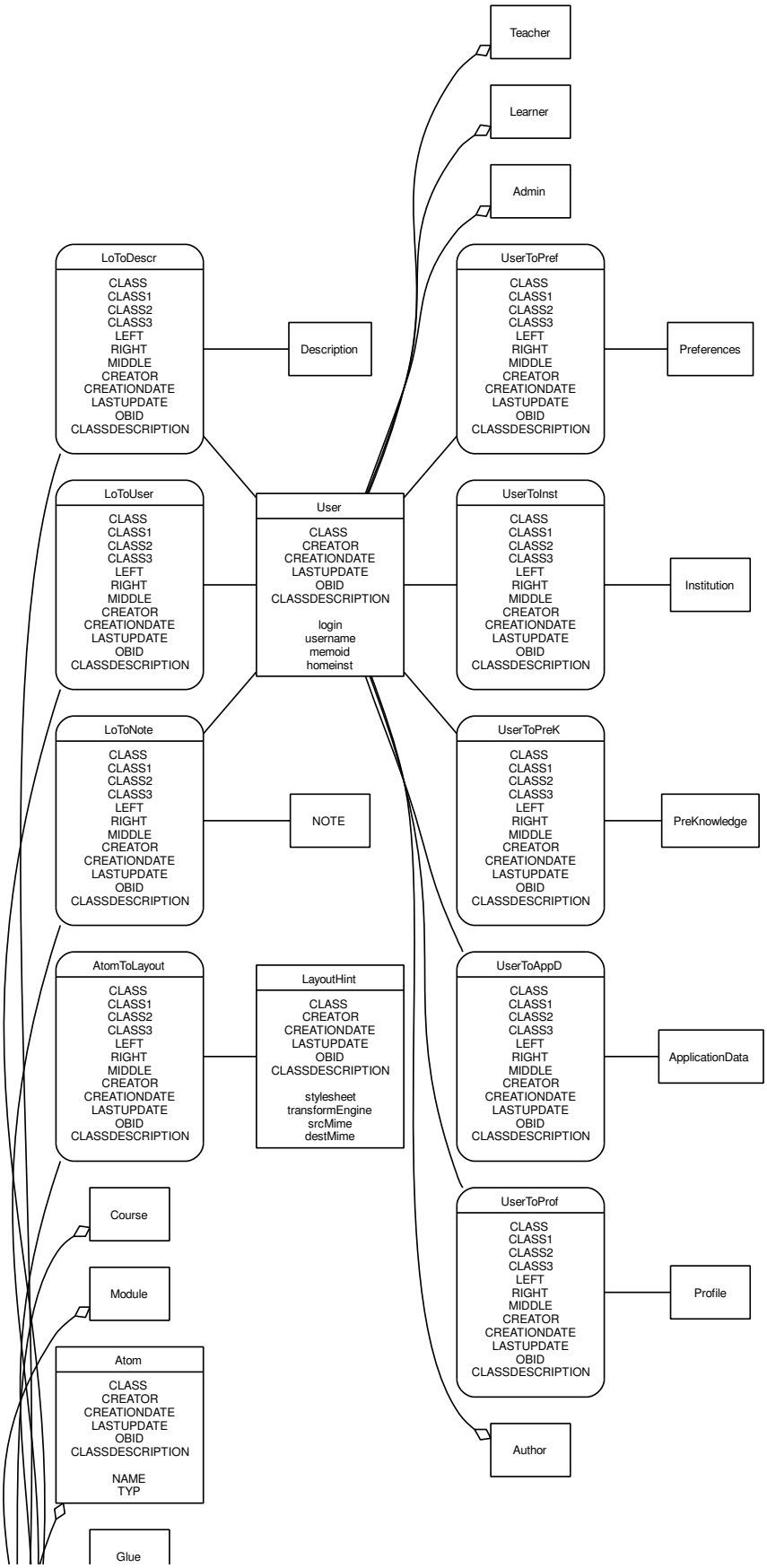


Abbildung D.1: Interne Datenstruktur des DLE-Systems

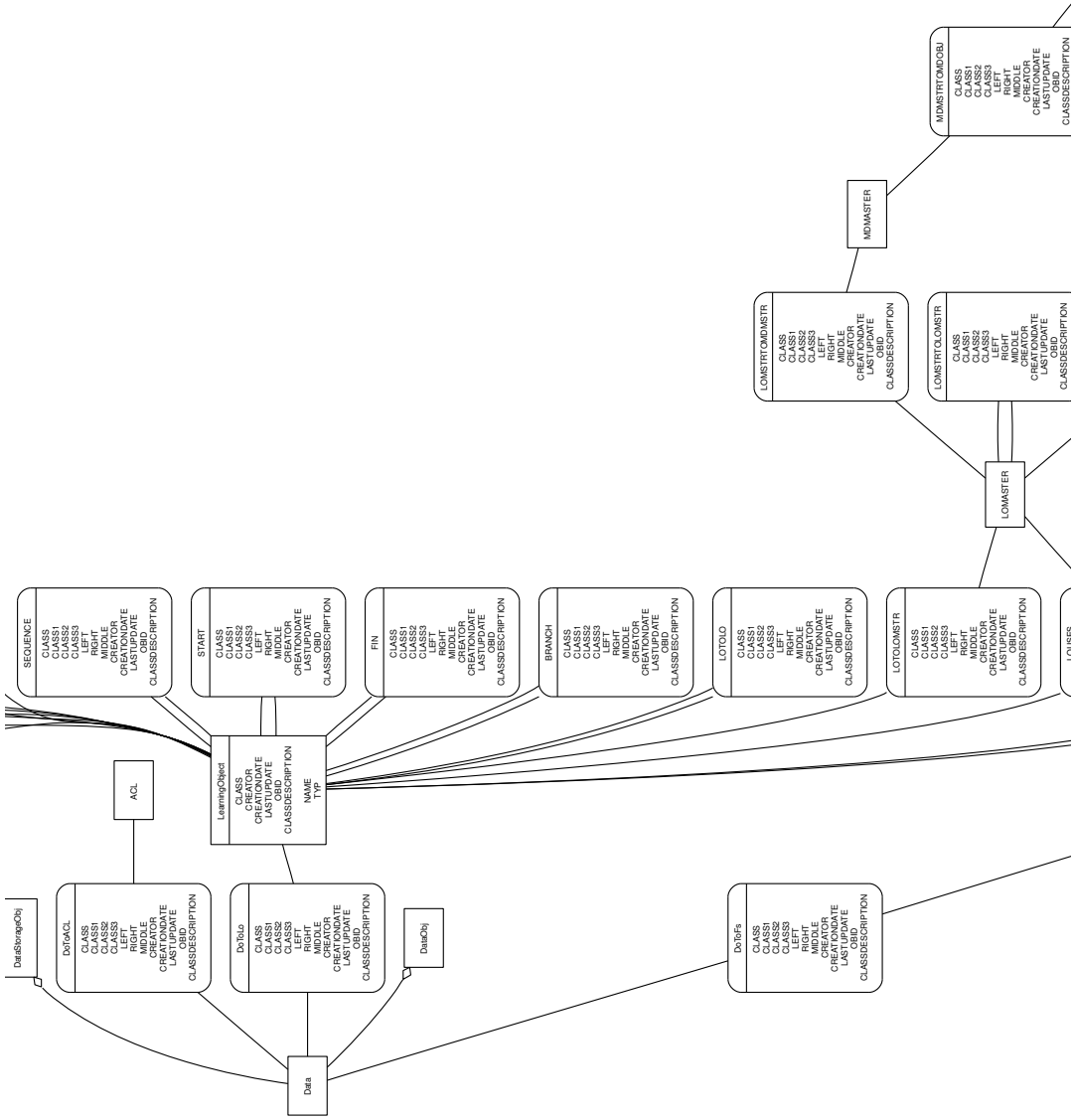


Abbildung D.2: Interne Datenstruktur des DLE-Systems

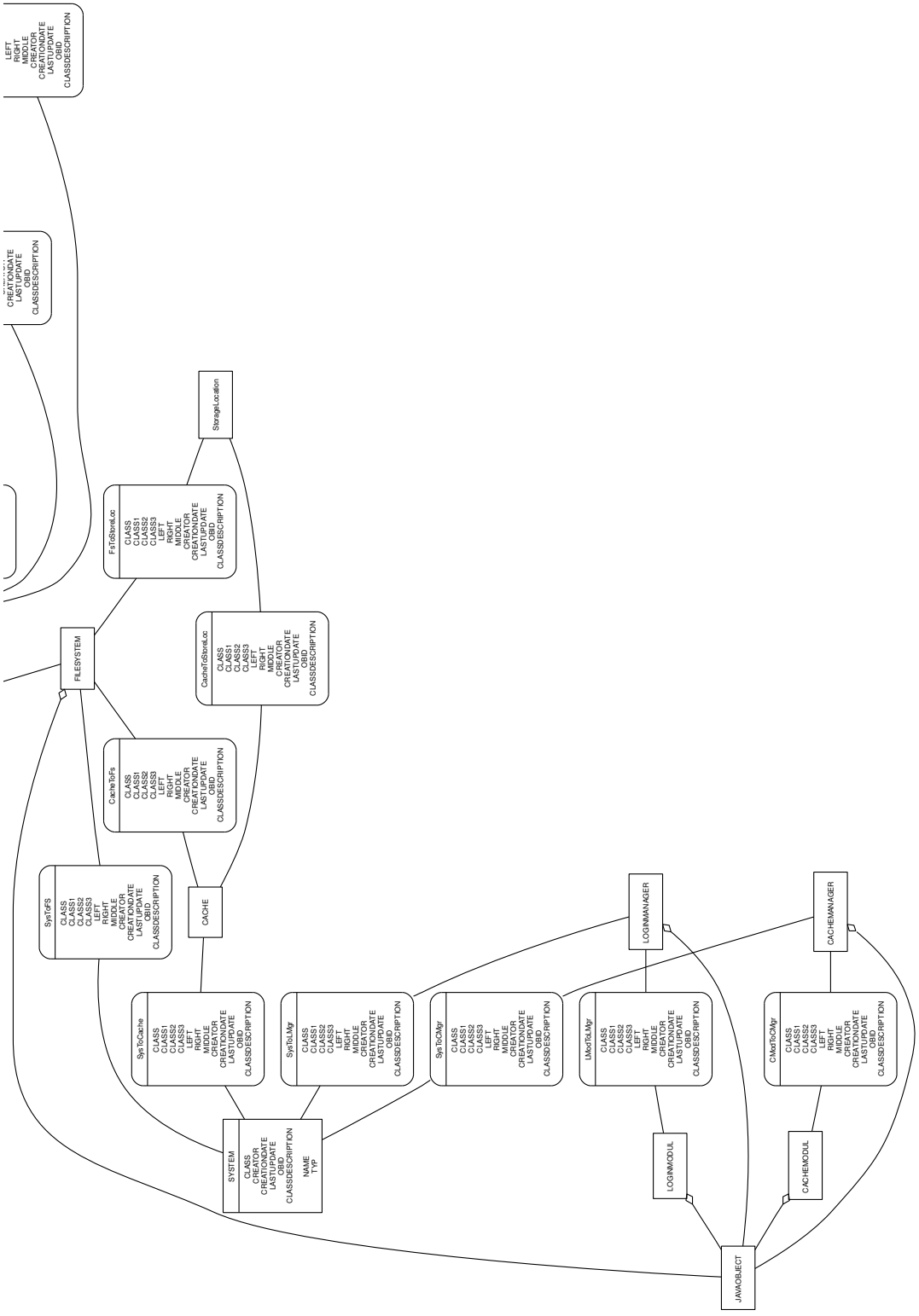


Abbildung D.3: Interne Datenstruktur des DLE-Systems

D.3 Klassenbaum der Simulationsbibliothek

Der dargestellte Klassenbaum der Simulations-Bibliothek gibt einen Überblick über die im Rahmen der vorliegenden Arbeit erstellten Simulationskomponenten und gibt einen Eindruck über die Verwendbarkeit der in Kapitel A.1 beschriebenen Bibliothek.

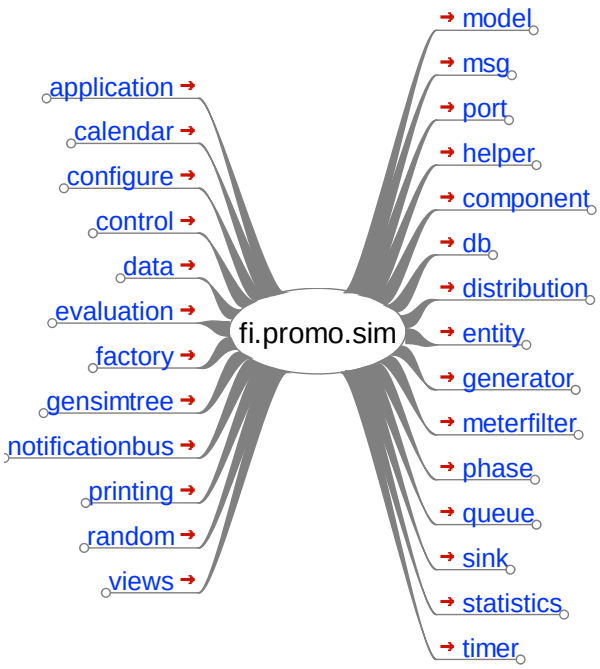


Abbildung D.4: Pakete des Simulations-Klassenbaumes

D.4 Strukturdefinition der Konfigurationsdatei der Simulationsbibliothek

```
<!ELEMENT Simulation (GeneralInformation,ParameterSet,(SimulationModel|Entity*),
                                                                    Evaluation) >
<!ELEMENT GeneralInformation (Title,SimulationTool,ResultsDirectory,
                                            (FixedParameter|RangeParameter)*) >
<!ELEMENT Title (#PCDATA) >
<!ELEMENT ParameterSet (FixedParameter|RangeParameter)*  >
<!ELEMENT FixedParameter (#PCDATA|List|Array|Matrix)* >
<!ATTLIST FixedParameter
     Name CDATA  #REQUIRED
    Class CDATA  #IMPLIED>

<!ELEMENT Evaluation  (Plot2D|Plot3D|Histplot)*  >
<!ELEMENT RangeParameter ((Value)+|ValRange)  >
<!ATTLIST RangeParameter
     Name CDATA  #REQUIRED >

<!ELEMENT  Value (#PCDATA) >
<!ELEMENT  ValRange (#PCDATA) >
<!ELEMENT SimulationTool (#PCDATA) >
<!ELEMENT ResultsDirectory (#PCDATA) >
<!ELEMENT List (Value)+ >
<!ELEMENT Array (CsvRow)+ >
<!ELEMENT Matrix (CsvRow)+ >
<!ELEMENT CsvRow (#PCDATA) >

<!ELEMENT Entity (ParameterSet,(Entity|Statistic|Distribution)*) >
<!ATTLIST Entity
     Name CDATA #REQUIRED
    Class CDATA #IMPLIED>

<!ELEMENT SimulationModel (ParameterSet,Entity*) >
<!ATTLIST SimulationModel
     Name CDATA #REQUIRED
    Class CDATA #IMPLIED>

<!ELEMENT Distribution (ParameterSet) >
<!ATTLIST Distribution
     Name CDATA #REQUIRED
     Type CDATA #IMPLIED>

<!ELEMENT Statistic (ParameterSet) >
<!ATTLIST Statistic
     Name CDATA #REQUIRED
     Type CDATA #IMPLIED>
```

Abbildung D.5: SimConfig DTD (Teil 1)

```
45 <!ELEMENT Plot2D (Title,(xLabel|yLabel|zLabel)*,(xRange|yRange|zRange)*,Param*,
46                                                                    xVar,yVar) >
47 <!ATTLIST Plot2D
48 generator CDATA #REQUIRED
49      size CDATA #IMPLIED
50   outname CDATA #IMPLIED
51       typ CDATA #IMPLIED>
52
53 <!ELEMENT Plot3D (Title,(xLabel|yLabel|zLabel)*,(xRange|yRange|zRange)*,Param*,
54                                                          xVar,yVar,zVar) >
55 <!ATTLIST Plot3D
56 generator CDATA #REQUIRED
57      size CDATA #IMPLIED
58   outname CDATA #IMPLIED
59       typ CDATA #IMPLIED>
60
61 <!ELEMENT HistPlot (Title,(xLabel|yLabel|zLabel)*,(xRange|yRange|zRange)*,Param*,
62                                                                       var) >
63 <!ATTLIST HistPlot
64 generator CDATA #REQUIRED
65      size CDATA #IMPLIED
66   outname CDATA #IMPLIED
67       typ CDATA #IMPLIED>
68
69 <!ELEMENT xLabel (#PCDATA) >
70 <!ELEMENT yLabel (#PCDATA) >
71 <!ELEMENT zLabel (#PCDATA) >
72 <!ELEMENT xRange (#PCDATA) >
73 <!ELEMENT yRange (#PCDATA) >
74 <!ELEMENT zRange (#PCDATA) >
75 <!ELEMENT Param (#PCDATA) >
76 <!ELEMENT var (#PCDATA) >
77 <!ELEMENT xVar (#PCDATA) >
78 <!ELEMENT yVar (#PCDATA) >
79 <!ELEMENT zVar (#PCDATA) >
80
81 <!ATTLIST var name CDATA #REQUIRED>
82 <!ATTLIST xVar name CDATA #REQUIRED>
83 <!ATTLIST yVar name CDATA #REQUIRED>
84 <!ATTLIST zVar name CDATA #REQUIRED>
85
86 <!ELEMENT TmpVar (#PCDATA)>
87 <!ATTLIST TmpVar name CDATA #REQUIRED>
```

Abbildung D.6: SimConfig (DTD Teil 2)