UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
(Institut für Informatik)

Dokumentenmanagement in kooperativen Arbeitsumgebungen

Diplomarbeit

Leipzig, September 2003

vorgelegt von

Michael Stach
geb. am 9. Juli 1977

Studiengang Informatik

Kurzzusammenfassung

Dokumentenmanagement war schon immer eine wichtige Aufgabe in verschiedenen Bereichen, insbesondere in Unternehmen und Behörden. Immer mehr Dokumente entstehen zu großen Teilen oder vollständig in elektronischer Form. Meist arbeiten mehrere Personen gemeinsam an Dokumenten. Elekronische Dokumentenmanagementsysteme müssen sich daher ganz gezielt diesen Gegebenheiten anpassen, um kooperative Arbeit zu unterstützen.

Die Diplomarbeit dokumentiert die Konzeptionierung eines Dokumentenmanagementsystems für kooperative Arbeitsumgebungen. Dabei wird auf sämtliche Belange der systematischen Arbeit mit Dokumenten eingegangen, sowohl in Hinsicht auf allgemeine Anforderungen an ein Dokumentenmanagement, als auch unter Berücksichtigung der speziellen Anforderungen der kooperativen Arbeit an den Dokumenten.

Basierend auf einer ausführlichen Anforderungsanalyse wird ein Dokumentenmanagementsystem entworfen, welches dem aktuellen Stand der Technik entspricht und auf Technologien wie Java und XML aufsetzt.

Danksagung

Ich möchte mich bei allen Personen, die mich bei der Arbeit an dieser Diplomarbeit unterstützt haben, bedanken. Mein besonderer Dank gilt Herrn Prof. Christian Wolff und Herrn Erik Wende, die mich nicht nur fachlich sondern auch moralisch unterstützten. Nicht vergessen möchte ich Herrn Hendrik Schubert und die eWerk IT GmbH, die mir die Arbeit an diesem interessanten Thema erst ermöglicht haben. Weiteren Dank schulde ich meinen Eltern, meinem Bruder und Herrn Ralf Müller, die mir stets mit gutem Rat zur Seite standen.

Inhaltsverzeichnis

Tabellenverzeichnis vii

Abbildungsverzeichnis viii

1 Einleitung 1
- 1.1 Umfeld der Diplomarbeit . . . 2
- 1.2 Dokumentenmanagement in Intranets . . . 2
- 1.3 Abgrenzung . . . 4

2 Anforderungsanalyse 7
- 2.1 Einleitung . . . 7
- 2.2 Anforderungen an Dokumentenmanagementsysteme . . . 7
  - 2.2.1 Lebenszyklus eines Dokuments . . . 7
  - 2.2.2 Strukturierung der Dokumente . . . 9
    - 2.2.2.1 Verbunddokumente . . . 11
    - 2.2.2.2 Versionierung . . . 12
  - 2.2.3 Organisation der Dokumente . . . 13
  - 2.2.4 Zugriff auf Dokumente . . . 15
- 2.3 Anforderungen in kooperativen Arbeitsumgebungen . . . 16
  - 2.3.1 Infrastruktur . . . 16
  - 2.3.2 Allgemeine Anforderungen in kooperativen Arbeitsumgebungen . 17
    - 2.3.2.1 Arbeit mit Dokumenten . . . 17
    - 2.3.2.2 Abbildung der Wirtschaftsprozesse . . . 20
    - 2.3.2.3 Adhoc-Workflows . . . 21
  - 2.3.3 Spezielle Anforderungen in kleinen Umgebungen . . . 22
  - 2.3.4 Spezielle Anforderungen in mittleren und großen Umgebungen . . 23
- 2.4 Notwendige Schnittstellen . . . 25
  - 2.4.1 Benutzerverwaltung . . . 25
  - 2.4.2 Suche im Dokumentenbestand . . . 26
  - 2.4.3 Unterstützung von Workflows . . . 26
  - 2.4.4 Datenaustausch mit dem Benutzer . . . 27
  - 2.4.5 Speicherung der Daten . . . 28
- 2.5 Akzeptanzkriterien für Dokumentenmanagementsysteme . . . 29
- 2.6 Anforderungsevaluierung bei Dokumentenmanagementsystemen . . . 30
- 2.7 Gründe für die Notwendigkeit einer Neuentwicklung . . . 30

3 Entwurf und Design **35**
3.1 Struktur des Softwaresystems . . . 35
3.1.1 Web-Client . . . 36
3.1.2 Dokumentenmanagementsystem-Kern . . . 36
3.1.3 Datenspeicher . . . 37
3.1.4 Dokumentenausgabe . . . 37
3.1.5 Beispiel . . . 38
3.2 Auswahl der Technologien . . . 38
3.2.1 Web-Client . . . 38
3.2.2 Java . . . 39
3.2.2.1 Servlets . . . 40
3.2.2.2 Enterprise JavaBeans . . . 40
3.2.2.3 JDBC . . . 41
3.2.3 WebDAV . . . 41
3.2.4 XML . . . 42
3.3 Kernsystem . . . 42
3.3.1 Aufbau . . . 42
3.3.2 Dispatcher . . . 44
3.3.2.1 Annahme von und Antwort auf Anfragen . . . 44
3.3.2.2 Verteilung der Anfragen . . . 44
3.3.2.3 Kommunikation mit den Modulen . . . 45
3.3.3 Task-Scheduling-Service . . . 45
3.3.4 Module . . . 45
3.3.4.1 Extraktion von Anfragen . . . 46
3.3.4.2 Speicherung von Einstellungen . . . 46
3.3.4.3 Reaktionsmöglichkeiten . . . 46
3.4 Kernmodule . . . 47
3.4.1 Kernfunktionen des Dokumentenmanagements . . . 47
3.4.2 Benutzerverwaltung und Authentifizierung . . . 49
3.4.2.1 Aufbau der Benutzerverwaltung . . . 49
3.4.2.2 Anbindung an externe Systeme . . . 50
3.4.3 Ein- und Ausgabe . . . 50
3.4.4 Datenspeicherung . . . 50
3.5 Funktionsmodule . . . 51
3.5.1 Groupware-Modul . . . 51
3.5.1.1 Übersicht zu den Funktionalitäten des Moduls . . . 51
3.5.1.2 Benutzergesteuerte Aktivitäten . . . 52
3.5.1.3 Zeitgesteuerte Aktivitäten . . . 52
3.5.2 Andere Module . . . 53
3.6 Benutzersicht . . . 53

4 Schnittstellen **58**
4.1 Schnittstellen nach außen . . . 58
4.1.1 XML-basierter Zugriff der Clients . . . 58

4.1.1.1 Anfragen . . . 58
4.1.1.2 Antworten . . . 60
4.2 Schnittstellen nach innen . . . 60
4.2.1 Zugriff auf den Datenspeicher . . . 61
4.2.2 Zugriff auf die Benutzerverwaltung . . . 63
4.2.3 Zugriff auf den Task-Scheduling-Service . . . 64
4.2.4 Erweiterungsmodule . . . 65
4.2.4.1 Klassenschnittstelle . . . 65
4.2.4.2 Registrierung des Moduls . . . 67
4.2.4.3 Definition der XML-Schemata . . . 68
4.2.5 Erweiterungsmodule für die Authentifizierung . . . 68
4.2.6 Erweiterungsmodule für Datenspeicher . . . 69
4.2.7 Erweiterungsmodule für Dokumentenein- und -ausgabe . . . 70

5 Ausblick **73**

Literaturverzeichnis **74**

A Funktionsübersicht **85**
A.1 Kernmodul des Dokumentenmanagements . . . 85
A.2 Modul zur Benutzerverwaltung . . . 90
A.3 Modul zur Datenspeicherung . . . 95
A.4 Modul für Groupware-Funktionalität . . . 96
A.4.1 Erweitertungen von Funktionen der Kernmodule . . . 96
A.4.2 Funktionen des Moduls . . . 98

B XML-Formate **104**
B.1 Schemata der XML-Kommunikation . . . 104
B.1.1 Wrapper . . . 104
B.1.2 Kernfunktionen . . . 104
B.1.3 Authentifizierung . . . 104
B.1.4 Benutzerverwaltung . . . 105
B.1.5 Ein- und Ausgabemodule . . . 105
B.1.6 Groupware-Modul . . . 105

C Quelltexte **106**
C.1 Zugriff auf Dokumente . . . 106
C.2 Zugriff auf Konfigurationen . . . 108
C.3 Modulschnittstelle für Benutzerspeicher . . . 109
C.4 Zugriff auf Benutzerdaten . . . 111
C.5 Modulschnittstelle . . . 114
C.6 Modulschnittstelle zur Authentifizierung . . . 115
C.7 Dispatcher . . . 116
C.8 Task-Scheduling-Service . . . 118

C.9 Modulschnittstelle für Datenspeicher 119
C.10 Modulschnittstelle für Eingabemodule 121
C.11 Modulschnittstelle für Ausgabemodule 123

Tabellenverzeichnis

2.1 Klassifizierung von Dokumenten nach Verlauf ihres Lebenszykluses 10
2.2 Klassifizierung von Dokumenten nach Inhalt 10
2.3 Typische Probleme des Dokumentenmanagements nach Art des Dokuments 11
2.4 Klassifizierung von Groupware nach Zeit und Ort 18
2.5 Klassifizierung von Groupware nach Art der Anwendung 32
2.6 Marktübersicht über Dokumentenmanagementsysteme (1/2) 33
2.7 Marktübersicht über Dokumentenmanagementsysteme (2/2) 34

Abbildungsverzeichnis

1.1 Oberfläche des CMS conrad 6

2.1 Lebenszyklus eines Dokuments 8
2.2 Zugriffshäufigkeit im Lebenszyklus eines Dokuments 9
2.3 Klassifikation nach dem 3C-Modell 19

3.1 Grobstruktur des Dokumentenmanagementsystems 36
3.2 Architektur des Dokumentenmanagementsystems 43
3.3 Interne Struktur eines Dokuments 48
3.4 Struktur eines Dokuments und Beispiel 56
3.5 Sicht des Benutzers auf das DMS 57

4.1 Beispiel für eine Anfrage an das DMS 59
4.2 Beispiel für eine Antwort des DMS 61
4.3 UML-Klassendiagramm der DocumentStorage-Schnittstelle des DMS . . . 62
4.4 UML-Klassendiagramm der ConfigurationStorage-Schnittstelle 62
4.5 UML-Klassendiagramm der Modulschnittstelle für Benutzerspeicher . . . 63
4.6 UML-Klassendiagramm des Benutzermanagements des DMS 64
4.7 UML-Klassendiagramm des Task-Scheduling-Service des DMS 64
4.8 Code-Fragment zum Zugriff auf den Task-Scheduling-Service 65
4.9 UML-Klassendiagramm der Modulschnittstelle des DMS 66
4.10 Beispiel für eine Modulimplementierung 67
4.11 UML-Klassendiagramm der Modulschnittstelle zur Authentifizierung . . . 69
4.12 UML-Klassendiagramm der Modulschnittstelle für Datenspeicher 69
4.13 UML-Klassendiagramm für Dokumentenein- und -ausgabe 71

1 Einleitung

Die Organisation der Dokumente, die in allen Bereichen – sei es nun in Wissenschaft oder Wirtschaft – entstehen, ist eine essentielle Aufgabe. Standardsoftware und Betriebssysteme unterstützen dabei nur mit einigen Grundfunktionen. Dokumente sind jedoch der wesentliche Speicherort für Wissen, und dieses Wissen muss zur effizienten Nutzung gut organisiert und sicher abgespeichert werden. In den seltensten Fällen arbeitet nur eine Person an einem Dokument. Sobald aber mehrere Personen auf ein Dokument zugreifen, muss es durch geeignete Maßnahmen geschützt werden, um konsistent zu bleiben. Die gemeinsame Arbeit an Dokumenten soll so einfach wie möglich sein. Zum Erreichen dieser Ziele existieren unterschiedliche Arten von Software, die jeweilige Teilaspekte abdecken, so zum Beispiel Groupware für die Kooperationsunterstützung und Dokumentenmangementsysteme zur Organisation der Dokumente selbst. Der Einsatz dieser Softwaresysteme geht mit hohen Kosten für die Lizensierung einher, so dass oft darauf verzichtet wird.

Ziel dieser Diplomarbeit ist es, ein Dokumentenmanagementsystem zu konzeptionieren, dass die Organisation der Dokumente einfach gestaltet, die Zusammenarbeit der Bearbeiter vereinfacht und an die jeweiligen Bedürfnisse angepasst werden kann. Zur Zielgruppe des Dokumentenmanagementsystems gehören nicht nur Firmen mit mehr als 50 Mitarbeitern und entsprechend ausgeprägten Hierarchien, sondern auch kleine Firmen, die bisher aus Kostengründen kein solches System einsetzen.

Diese Arbeit umfasst eine Anforderungsanalyse (Kapitel 2), die einerseits die potentiellen Kunden des Systems beschreibt und andererseits die notwendige Funktionalität und deren Prinzipien aufführt. In Kapitel 3 wird die Phase des Entwurfs mit den darin gefällten Entscheidungen umfassend dokumentiert und der Kontext der Entscheidungsfindung dargestellt. In Kapitel 4 sind die Schnittstellen des Systems nach innen und außen dokumentiert, und es wird beispielhaft auf die Programmierung und den Einsatz des Dokumentenmanagementsystems eingegangen. Abschließend wird in Kapitel 5 ein Ausblick auf die mögliche (und notwendige) Weiterentwicklungen des Systems gegeben.

1.1 Umfeld der Diplomarbeit

Diese Diplomarbeit entstand in der eWerk IT GmbH. Das zu konzeptionierende Dokumentenmanagementsystem wurde in die vorhandene Produktpalette eingepasst.

Die eWerk IT GmbH ist ein in Leipzig ansässiges New-Media-Unternehmen mit Außenstellen in Berlin und Hamburg. Bereits seit 1994 am Markt, gehört die Firma mit 60 Mitarbeitern, darunter 20 Festangestellten, und über 1500 kleineren und großen Kunden jeglicher Branchen zu den führenden IT-Beratungsunternehmen im Raum Leipzig.

Das Leistungsspektrum ist sehr breit gefächert und beinhaltet als Schwerpunkte Unternehmens-, Technologie- und Prozessberatung. Die eWerk IT GmbH agiert darüber hinaus als Systemhaus mit über 200 Produkten, Agentur für CGM- und Multimediaservices, Soft- und Hardwareservices sowie als Rechenzentrum an vier verschiedenen Standorten.

Zu den bereits vorhandenen Produkten gehört das Content- und Assetmanagementsystem conrad, das sowohl in Intra- als auch in Internetlösungen zum Einsatz kommt. Mit einer vollständigen Bedienbarkeit nach dem Prinzip des WYSIWYG (What You See Is What You Get) und strikter Trennung von Inhalt und Layout kommt es bereits bei mehreren Kunden erfolgreich zum Einsatz. Es hat unter anderem Schnittstellen zu Lotus Domino, SAP, LDAP und Active Directory und befindet sich ständig in Weiterentwicklung. In Zukunft soll conrad stark in Richtung Enterprise Content Management erweitert werden, was auch die Integration einer Dokumenten- und Workflowmanagementkomponente beinhaltet. Dem entsprechend dokumentiert diese Diplomarbeit wichtige Entwicklungsarbeit in diesem Tätigkeitsfeld der eWerk IT GmbH.

1.2 Dokumentenmanagement in Intranets

Dokumentenmanagement als Begriff kann sehr weit gefasst werden. Das schließt dabei sowohl Papier- als auch elektronische Dokumente ein. Der Begriff Dokumentenmanagement kann ähnlich wie in [1] allgemein definiert werden:

> Dokumentenmanagement ist eine systematische Methode zum Speichern, Auffinden von und Übersicht behalten über Dokumente.

Der Begriff Dokument könnte dann wie folgt definiert werden:

> Als Dokument versteht man eine Menge von kodierter Information in materieller Form.

Diese Definition ist noch recht vage und allgemein gehalten. Ein Dokument kann demnach vieles sein. Beispiele wären etwa eine Personalakte, eine Projektmappe, eine Graphik, eine elektronische Textdatei, ein Notizblatt oder ein digitalisierter Geschäftsbrief.

Im Rahmen dieser Diplomarbeit ist es jedoch günstiger, die Begriffe enger zu fassen. Generell sollen nur elektronische Dokumente betrachtet werden. Liegen Dokumente nicht in elektronischer Form vor, so können diese digitalisiert werden. Es ergibt sich also die Notwendigkeit eines elekronischen Dokumentenmanagementsystems, das heißt einer Software, die – laut obiger Definition – die elektronischen Dokumente systematisch speichert und auffinden kann. Im folgenden sind unter Dokumenten und Dokumentenmanagementsystemen immer elektronische Dokumente und Dokumentenmanagementsysteme (DMS) zu verstehen.

Bei der elektronischen Verwaltung von Dokumenten ist der Begriff der Datei ein zentrales Element. Betriebssysteme verwenden dieses Konzept. Eine Datei ist ein Speicherbereich, der beliebige binäre Daten fassen kann und dem Benutzer gegenüber als kleinstes logisches Element bei der Speicherung von Daten durch ein Betriebssystem verwendet wird. Solche Dateien lassen sich in Ordner oder Verzeichnisse einsortieren. Diese Hilfsmittel ermöglichen ein einfaches Dokumentenmanagement. Problematisch wird diese Methode erst, wenn mehrere Personen parallel auf den Dokumentenbestand zugreifen müssen beziehungsweise die Anzahl der Dokumente ein bestimmtes Maß überschreitet. Dann wird ein komplexeres DMS benötigt, um einen effizienten Umgang mit Dokumenten zu gewährleisten. Es ist davon auszugehen, dass der Einsatz eines einheitlichen DMS erst ab einer bestimmten Anzahl von Dokumenten beziehungsweise einer bestimmten Anzahl von Benutzern einen Vorteil bringt. Darum soll im folgenden unter einem DMS immer ein solches komplexes elektronisches System verstanden werden, das im Gegensatz zum einfachen Dokumentenmanagement mit Dateien und Verzeichnissen auf einem Computer wesentlich komplexere Methoden zur Verwaltung verwendet.

Die zwei elementaren Tätigkeiten im Umgang mit allen Dokumentenmanagementsystemen sind also das Ablegen und Auffinden von Dokumenten. In modernen Systemen sind diese beiden Grundfunktionen mit sehr großer Detailgenauigkeit und Konfigurierbarkeit implementiert, so dass alle Aspekte des Lebenszykluses eines Dokuments (siehe Abschnitt 2.2.1) von der Erstellung und Bearbeitung bis hin zur Archivierung abgedeckt werden können.

Bei einem solchen Funktionsumfang ist der Einsatz wiederum nur in größeren Strukturen wie Firmen, öffentlichen Einrichtungen oder Organisationen sinnvoll. In diesen

Anwendungsfällen ist meistens eine Netzwerkinfrastruktur vorhanden, die die einzelnen Arbeitsplätze und damit die Personen, die die Dokumenten des Unternehmens erstellen, auf elektronischem Wege verbindet. Dieses Netzwerk wird als Intranet bezeichnet und ist nur Mitarbeitern des jeweiligen Unternehmens zugänglich.

Fast jedes Unternehmen ist in mehrere Arbeitsgruppen oder Abteilungen aufgegliedert, die sowohl an eigenen als auch an gemeinsamen oder unternehmensweit relevanten Dokumenten arbeiten. Software, die dieses gemeinsame Arbeiten unterstützt, wird als Groupware bezeichnet. Eng damit verbunden ist der Begriff der Computer Supported Cooperative Work (CSCW) [2, S. 88–89], die Bezeichnung des Forschungsgebiets, das die gemeinsame Arbeit mittels Computer untersucht. Ein einfaches CSCW-System ist zum Beispiel in [3] beschrieben. Groupware- bzw. CSCW-Lösungen sind oft in die Intranets integriert.

Ein Intranet ist ein Netzwerk, welches eine geschlossene Menge an Client-Rechnern mittels Standardinternetprotokollen (etwa TCP/IP [4] [5], HTTP [6]) verbindet [7]. Intranets werden in Unternehmen zum Informationsaustausch und zur gemeinsamen Arbeit verwendet.

Eine weitere Aufgabe einer Intranetlösung ist das Dokumentenmanagement. Deshalb macht es Sinn, die Teile der Groupware- und CSCW-Funktionalitäten in das Dokumentenmanagementsystem zu integrieren. In dieser Diplomarbeit soll ein DMS konzeptioniert werden, welches genau diese Anforderung erfüllt und die gemeinsame Arbeit von Gruppen an einem Dokument entsprechend unterstützt.

1.3 Abgrenzung

Die Diplomarbeit umfasst die Phasen der Anforderungsanalyse, des Entwurfs und Designs der Architektur des Dokumentenmanagementsystems sowie die Dokumentierung der Schnittstellen des entstehenden Systems.

Dabei beschränkt sich der Umfang auf die Grundfunktionalität und die Architektur des Systems und der Kooperationunterstützung innerhalb des Dokumentenmanagements. Die Grundfunktionalität umfasst das Verwalten, insbesondere das Speichern und Bearbeiten, der Dokumente. Die Kooperationunterstützung beinhaltet sämtliche Funktionen, die das gemeinsame Erstellen von Dokumenten in Intranets und anderen kooperativen Umgebungen unterstützen, und in den Bereich der Groupware beziehungsweise des CSCW eingeordnet werden können.

Nicht näher betrachet werden komplexere Funktionen zur Suche im Dokumentenbestand, die Realisierung von Workflows und deren Verwaltung und die Details der Speicherung der Binärdaten in geeigneten Systemen, wie etwa einem Datenbankmanagementsystem.

Es wird ebenfalls nicht Teil der Diplomarbeit sein, die Erstellung einer benutzerseitigen Anwendung zu dokumentieren, welche für den Einsatz des Systems notwendig sein wird. Lediglich auf einzelne Aspekte der Realisierung einer solchen Anwendung soll eingegangen werden, insbesondere auf die Usability-relevanten Bestandteile.

Diese Einschränkungen sind notwendig, da der Umfang einer vollständigen Dokumentation den Rahmen dieser Arbeit sprengen und deutlich mehr Zeit in Anspruch nehmen würde. Daher werden die Aspekte der Erstellung des Systems ausgelassen, die rein technischer Natur sind oder sich hauptsächlich mit anderen Themengebieten als dem Dokumentenmanagement beschäftigen.

Abbildung 1.1: Web-Oberfläche des Contentmanagementsystems conrad, Quelle: eWerk IT GmbH

2 Anforderungsanalyse

2.1 Einleitung

In diesem Abschnitt werden die Anforderungen an das zu konzeptionierende Dokumentenmanagementsystem (im folgenden auch kurz DMS) beschrieben. Dabei sind allgemeine Anforderungen, die für jedes Softwaresystem gelten, genauso einbezogen, wie Anforderungen an ein DMS, wie sie von Kunden an die eWerk IT GmbH herangetragen wurden. Sämtliche Überlegungen sind auf die angestrebte Zielgruppe des DMS ausgerichtet.

Zuerst sollen allgemeine Anforderungen an ein Dokumentenmanagement (Abschnitt 2.2) zusammengefasst werden, danach werden sie in Abschnitt 2.3 auf das spezielle Umfeld kooperativer Arbeitsumgebungen erweitert. Anschließend wird in Abschnitt 2.4 auf die benötigten Schnittstellen zum Benutzer und zu anderer Software eingegangen. Es folgen einige Ausführungen zu allgemeinen Problemen bei der Einführung neuer Softwaresysteme (Abschnitt 2.5) und eine Übersicht über am Markt vorhandene DMS (Abschnitt 2.6) sowie eine Zusammenfassung, warum eine Neuentwicklung eines DMS notwendig erscheint (Abschnitt 2.7).

Am Ende dieses Kapitels wird der Grundstein für den Entwurf und das Design, Kapitel 3, gelegt. Damit ist dieses Kapitel auch im Sinne eines Lastenhefts zu sehen.

2.2 Anforderungen an Dokumentenmanagementsysteme

2.2.1 Lebenszyklus eines Dokuments

In Abschnitt 1.2 wurden bereits die Begriffe Dokument und Dokumentenmanagement abgesteckt und entsprechend die beiden Hauptaufgaben eines Dokumentenmanagementsystems genannt: Ablage und Auffinden von Dokumenten. Um jedoch die Anforderungen an ein Dokumentenmanagement genauer beschreiben zu können, soll zunächst der typische Lebenszyklus eines Dokuments zusammengefasst werden.

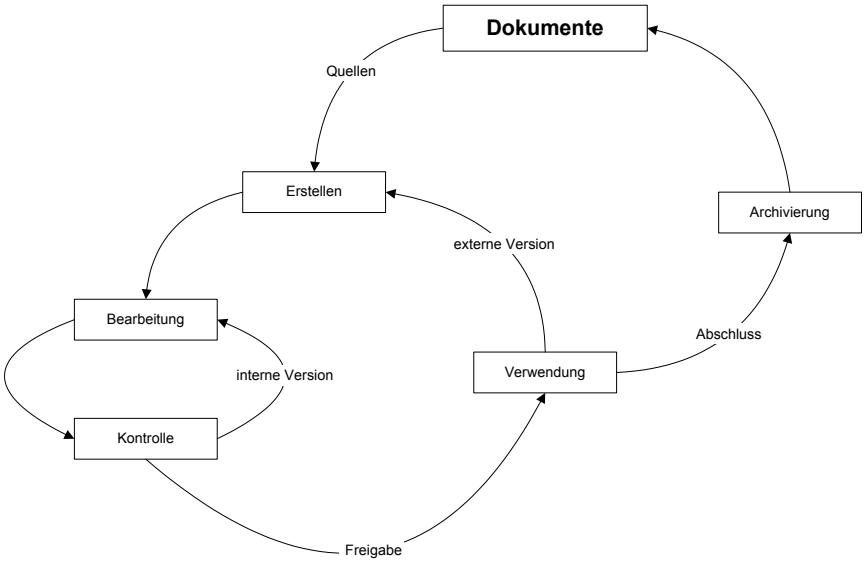

Abbildung 2.1: Lebenszyklus eines Dokuments, Quelle: Eigene Darstellung

Dokumente können sehr unterschiedliche Lebensdauern haben. Während einige Dokumente schon nach wenigen Stunden nicht mehr benötigt werden, können andere Dokumente über Jahre hinweg relevant sein und müssen dem entsprechend lange aufbewahrt werden. Zu der ersten Gruppe gehören Dokumentklassen wie Notizen, Nachrichten, Tonaufzeichnungen von Gesprächen, die danach in ein schriftliches Protokoll umgewandelt werden. Zu letzterer Gruppe gehören etwa Akten in Behörden, die selbst nach Jahren noch als Referenz gebraucht oder auch erweitert werden.

Der Lebenszyklus eines Dokuments – unabhängig von seiner Dauer – läuft jedoch in den meisten Fällen gleich ab [8]. In Abbildung 2.1 ist dieser Zyklus dargestellt. Die Zugriffshäufigkeit für Dokumente deren Lebensdauer einige Monate bis Jahre beträgt, ist in Abbildung 2.2 zu sehen. Sie ähnelt größtenteils kurzlebigeren Dokumenten. Zu Beginn erstellt eine Person das Dokument und erarbeitet es meistens mit Hilfe anderer Bearbeiter bis es einen Endstand erreicht hat. Beispielsweise wird ein Pflichtenheft vom Projektleiter angelegt und dann gemeinsam mit den Softwareentwicklern der Firma und eventuell unter Zuarbeit externer Berater erarbeitet. Hier ist eine sehr hohe Verwendungsfrequenz üblich. Nach der Erarbeitungsphase ist das Dokument abgeschlossen und wird verwendet (zweite Phase). Im Beispielfall wird die Software unter Zuhilfenahme des Pflichtenhefts erstellt bis sie vom Auftraggeber abgenommen wird. Dabei nimmt die Anzahl der Zugriffe auf das Dokument ab. Nach dem Abschluss der Arbeit an dem Projekt beginnt die dritte Phase, in der das Dokument nur selten als Referenz verwendet wird,

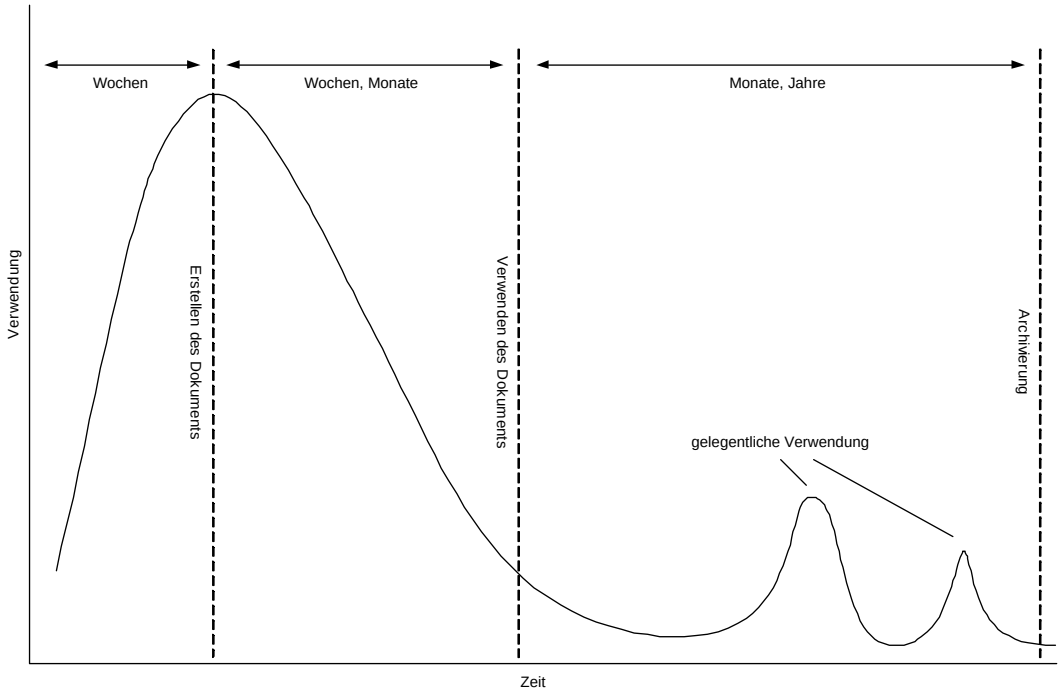

Abbildung 2.2: Zugriffshäufigkeit im Lebenszyklus eines Dokuments nach [8]

zum Beispiel zum Nachschlagen früher verwendeter Methoden, wenn neue Pflichtenhefte erstellt werden sollen.

In der dritten Phase sind die meisten Dokumente so selten in Gebrauch, dass sie üblicherweise nicht mehr sofort zugreifbar sein müssen und in Archiven gespeichert werden. Auch sind viele Bestandteile der Dokumente, wie etwa die Entstehungsgeschichte, nicht mehr relevant und müssen nicht mehr gespeichert werden. Unter Umständen können auch gesetzliche Bestimmungen zur maximalen oder minimalen Aufbewahrungsdauer existieren, etwa bei steuerrelevanten Rechnungen, die 10 Jahre aufbewahrt werden müssen [9], oder bei Abrechnungsdaten, die aus Gründen des Datenschutzes [10] nur über gewisse Zeiträume gespeichert bleiben dürfen [11]. Für bestimmte Dokumentklassen wie beispielsweise Notizen, Gesprächsaufzeichnungen oder ähnlich geartete Dokumente ist keine Speicherung über längere Zeit sinnvoll.

2.2.2 Strukturierung der Dokumente

Dokumente sind in ihrer Struktur sehr unterschiedlich. Grundsätzlich kann man viele verschiedene Arten von Dokumenten unterscheiden. Dokumente können nach Art der

Klasse	Beschreibung
Notiz	kurze Lebensdauer; wird bereits nach kurzer Zeit und einmaliger Verwendung gelöscht
Akte	nach Vorgang geordnete Sammlung von Daten; über eine lange Zeit kommen Daten weniger Bearbeiter hinzu; nach Abschluss der Bearbeitung wird die Akte aufbewahrt
Pflichtenheft	Beschreibung der zu leistenden Arbeit für einen Auftrag; Basis für neue Dokumente; Hohe Frequenz von lesenden Zugriffen

Tabelle 2.1: Klassifizierung von Dokumenten nach Verlauf ihres Lebenszykluses

Verwendung und nach Art des Inhalts in bestimmte Gruppen eingeteilt und unterschieden werden. In den Tabellen 2.1 und 2.2 werden nicht-vollständige Listen solcher Dokumentklassen gegeben, während in Tabelle 2.3 typische Probleme einiger Dokumentarten aufgelistet werden. Dabei sind insbesondere bei der Klassifizierung nach Inhalt auch Mischformen vorhanden, etwa Dokumente aus dem Bereich des Desktop-Publishing mit Text- und Bildinformationen.

Da in den meisten Unternehmen eine Vielzahl solcher Dokumentklassen verwendet wird, muss ein Dokumentenmanagementsystem alle Dokumentklassen unterstützen. Dabei müssen einige für diese Dokumentklassen spezifischen Eigenschaften bedacht werden. So sind etwa Videodaten sehr speicherintensiv (mehrere Gigabyte), oder Tonaufzeichnungen benötigen eine genaue Beschreibung des Inhalts, da zur Bestimmung des Inhalts sonst das Dokument zeitaufwändig abgespielt werden müsste.

Metadaten sind in den meisten Fällen für ein effizientes Dokumentenmanagement notwendig, insbesondere für das Auffinden von Dokumenten. Es gibt mehrere verschiedene Standards für Metadaten wie Dublin Core [12] oder anwendungsspezifische Metadaten, die aber alle unterstützt werden sollten. Die wichtigsten Metadaten sind Titel und Autor(en) eines Dokuments, aber auch seine Entstehungsgeschichte (Bearbeiter, Daten der Änderung etc.).

Klasse	Beschreibung/Beispiele
Text	Textverarbeitung, Programmierung (Office, Quelltexte, etc.)
Bild	Graphikbearbeitung, Druck
Audio	Aufzeichnungen von Gesprächen, Diktate, Tonaufzeichnungen
Video	Film- und TV-Produktionen, Website-Design

Tabelle 2.2: Klassifizierung von Dokumenten nach Inhalt

Klasse	Probleme
Video	– hohes Datenvolumen – nicht-textuelle Information – zeitabhängig
DTP	– hohes Datenvolumen – teilweise nicht-textuelle Informationen – viele Abhängigkeiten unter den Dateien
Audio	– mittleres bis hohes Datenvolumen – nicht-textuelle Information – zeitabhängig
Softwareentwicklung	– viele Bearbeiter – viele Abhängigkeiten unter den Dateien
Büroanwendungen	– teilweise nicht-textuelle Information – teilweise sehr kurzlebig

Tabelle 2.3: Typische Probleme des Dokumentenmanagements nach Art des Dokuments

2.2.2.1 Verbunddokumente

Die Struktur eines Dokuments gleicht nicht immer seiner Repräsentation im Computer. Es ist keine eindeutige Zuordnung von Datei zu Dokument möglich, da in den seltensten Fällen ein Dokument durch eine einzelne Datei darstellbar ist. Dies funktioniert zwar für einfache Dokumente wie Notizen oder Briefe, aber im Bereich des Desktop-Publishing (DTP) bestehen Dokumente aus dem Textsatz, den verwendeten Bildern und Schriftarten, die alle in einzelnen Dateien vorliegen. Die Arbeit wird gerade im Bereich des DTP durch die mangelnde Unterstützung für Dokumente, die aus mehreren Dateien bestehen, in vielen Dokumentenmanagementsystemen erschwert [13] [14].

Dokumente, die aus mehreren Dateien bestehen, werden als Verbunddokumente (compound documents) bezeichnet. Dabei wird ein Verzeichnis in einem Dateisystem als ein Dokument angesehen und dessen Dateien und Unterverzeichnisse bilden den Inhalt des Dokuments.

Der Begriff des Dokuments kann sehr weit oder sehr eng gefasst werden. Natürlich können auch Teile eines Dokuments als Dokument gesehen werden. In den meisten Fällen ist eine weite Fassung des Begriffs praktischer, da die Arbeit mit dem Dokument einfacher wird. Ein gutes Dokumentenmanagementsystem sollte beide Herangehensweisen unterstützen.

2.2.2.2 Versionierung

Eine weitere wichtige Funktion eines DMS ist die Versionierung. Unter Versionierung wird das Bereithalten aller (sinnvollen) Bearbeitungsstände eines Dokuments verstanden. Nach jeder Änderung des Dokuments wird eine neue Version gespeichert und die vorhergehende Fassung bleibt unverändert gespeichert. Unter Änderung ist hier eine vollständige Bearbeitung zu verstehen, also das Holen aus dem DMS, die Änderungen an den entsprechenden Dateien des Dokuments und das Zurückschreiben in das DMS.

Eine Versionierung ist insbesondere bei der Softwareentwicklung üblich, wo die Quellcodes der Software und deren Dokumentation mit Systemen wie CVS [15], SCCS [16] oder RCS [17] dateibasiert versioniert und verwaltet werden. Aber auch im Dokumentenmanagement ist eine Versionierung sinnvoll, um beispielsweise Arbeitsstände eines Dokuments (zu einem bestimmten Datum) rekonstruieren zu können.

Anders als bei der Softwareentwicklung ist es aber weder notwendig noch möglich im Dokumentenmanagement den vollen Umfang der Versionierungsfunktionalität der oben genannten Systeme umzusetzen.

Es ist im Allgemeinen nicht möglich, zwei Versionen, die eine gemeinsame Vorgängerversion haben, automatisch abzugleichen und eine einzige neue Version zu erstellen, da die möglichen Dateitypen dies nicht immer zulassen – etwa Videodaten – und der Aufwand für eine breite Unterstützung einfach zu hoch wäre. Außerdem ist eine solche Funktion nur bei wenigen Dokumentarten überhaupt sinnvoll (Quelltexte für Software, Textdateien). Bei graphischen Daten (Konstruktionspläne aus dem CAD/CAM-Bereich oder Bilddaten aus dem Desktop Publishing) lassen sich Versionen nicht automatisch zusammenführen. Einer der Gründe ist die komplexe Struktur der Daten, die nicht automatisch erfasst werden kann, was aber notwendig wäre, um die Änderungen zwischen zwei Fassungen zu identifizieren. Bei Quelltexten oder Textdateien sind solche Mechanismen in Programmen wie den diffutils [18] implementiert, die vom CVS-System unter anderem dazu genutzt werden, nur die Änderungen an einer Dateiversion zu speichern. Diese Änderungen können mit patch [19] dann wieder angewendet werden, unter Umständen auch auf Dateien, die nicht der ursprünglichen Fassung entsprechen, da das System zeilenbasiert arbeitet. Es existieren zwar auch Systeme, die die Änderungen in Binärdateien identifizieren und ähnlich diffutils und patch verwalten können [20] [21], aber sie funktionieren nicht bei geänderten Dateien, was jedoch für eine Zusammenführung zweier Versionen notwendig wäre.

Unter Anbetracht der breiten Benutzergruppen für ein DMS dürfen die Anforderungen an den Anwender nicht zu hoch sein. Die Versionierung muss daher weitgehend automatisch und ohne Zutun des Anwenders durchgeführt werden, so dass auch weniger erfahrene Benutzer die Versionierung nutzen können oder sie gegebenenfalls nicht – als störend – bemerken.

Um die Bedeutung einer Dokumentversion besser und schneller erfassen zu können, sollten die vorgenommenen Änderungen kommentiert werden können. Damit ist das aufwändige Durchlesen/Anschauen des Dokuments unnötig. Eventuell ist es dem Benutzer oder Bearbeiter des Dokuments auch gar nicht möglich, die Änderungen zu erkennen, beispielweise bei Videodaten mit minimalen Unterschieden. Solche Kommentare sind Teil der Metadaten eines Dokuments und ermöglichen eine effiziente Verwendung des Dokuments.

2.2.3 Organisation der Dokumente

Ein DMS wird hauptsächlich dazu genutzt, relevante Dokumente zu finden. Daher ist die strukturierte und gut organisierte Abspeicherung besonders wichtig. Das DMS soll das Auffinden von Dokumenten so einfach wie möglich machen. Dazu gibt es eine Vielzahl von Möglichkeiten, die Dokumente über Indizes und Metadaten zu ordnen und zu kategorisieren.

Das einfachste und bekannteste System ist die Einordung in Verzeichnisse wie in einem Dateisystem, in dem für jede Art von Dokumenten eine Verzeichnisstruktur vorhanden ist, worin systematisch eingeordnet werden kann. Im Vergleich zu Dateisystemen lässt sich eine Mehrfacheinordnung von Dokumente in verschiedene Verzeichnisse besser realisieren. Dabei muss diese Verzeichnisstruktur flexibel sein, also das Anlegen neuer Verzeichnisse, Untergliederungen (Unterverzeichnisse) und das Umsortieren von Dokumenten darin ermöglichen. Eine Erstellung eines solchen Systems kann unter Umständen sehr schwierig sein, da Dokumente an mehrere Stellen passen können oder nur schwer einzuordnen sind. Diese Verzeichnisse können sehr schnell anwachsen und benötigen dann aufwändige Pflege, um ihren Nutzen zu erhalten. Dies zeigt sich etwa bei Bibliotheken, die sehr umfangreiche Verzeichnisse führen [22], oder Internetverzeichnissen wie yahoo [23] oder web.de [24].

Eine weitere einfache Möglichkeit der Unterstützung der Suche nach Dokumenten ist die Beschlagwortung. Die Dokumente werden dazu mit Stichworten verknüpft, die den Inhalt des Dokuments beziehungsweise dessen Verwendung beschreiben. Diese Beschlagwortung

ist ebenfalls aus Bibliotheken bekannt. Problematisch ist auch hier der Aufbau eines Vokabulars für die Beschlagwortung und die konsequente Einhaltung dieses Vokabulars, da die Verwendung von Synonymen das Auffinden eines Dokuments verhindern könnte. Für eine große Anzahl von Dokumenten ist diese Aufgabe recht arbeitsaufwändig, kann aber durch Systeme zur automatischen Gewinnung von Metadaten (Stichworten) und über Thesauren unterstützt werden [25].

In ähnlicher Art und Weise können Dokumentgattungen (document genres) zur Organisation beitragen. Eine Dokumentgattung ist eine Gruppierung von Dokumenten mit ähnlicher Form oder Inhalt [26]. Die Gattungen, die verwendet werden, ergeben sich aus dem Umfeld des Einsatzes des DMS [27]. So könnten bespielsweise Quartalsbericht, Fehlerprotkoll, Memo etc. zu den Gattungen gehören. Die Gattungen selbst können wiederum in Untergattungen untergliedert oder mit anderen Gattungen verwandt sein [28], so dass sich schnell eine komplexe Struktur ergibt. Problematisch dabei ist, dass es in den seltensten Fällen möglich ist, sämtliche in einem Umfeld vorhandenen Genres durch eine Person zu erfassen zu lassen [29] und es sich um ein dynamisches System handelt [30]. Entsprechend muss ein DMS auf diese Situation vorbereitet sein.

Für große Mengen von Textdokumenten eignet sich eine Volltextindizierung. Dazu werden die in dem Text vorkommenden Worte in einem Index erfasst. Es kann dann nach Dokumenten gesucht werden, die bestimmte Worte enthalten [31]. Bekannte Beispiele sind Internetsuchmaschninen wie Google [32] und AltaVista [33]. Eine Volltextindizierung ist aber für viele Dokumente nicht anwendbar, etwa für Bilder, Video und Audio, kann also nur für textbasierte Dokumente genutzt werden.

Da Dokumente selten einzeln verwendet werden, geben die Relationen zu anderen Dokumenten die Möglichkeit, relevante Dokumente aufzufinden. Beispielsweise basiert ein Pflichtenheft auf einem Lastenheft, so dass, wenn diese Information gespeichert ist, es einfach ist, das Lastenheft zu finden, um das Pflichtenheft zu überarbeiten. Ein anderes Beispiel sind Literaturverweise, worüber man schnell zu einem Thema relevante Literatur finden kann, zum Beispiel bei der Suchmaschine CiteSeer [34].

Ein gutes Dokumentenmanagementsystem sollte möglichst viele der oben genannten Möglichkeiten der Organisation unterstützen, insbesondere, da die Effizienz der verschiedenen Methoden je nach Dokumentenklasse differiert.

2.2.4 Zugriff auf Dokumente

An die Möglichkeiten zum Zugriff auf die Dokumente in einem DMS sind zentrale Bedingungen geknüpft. Am wichtigsten ist der Bereich des Datenschutzes beziehungsweise der Datensicherheit.

Wichtigstes Mittel zur Zugriffskontrolle sind Benutzerberechtigungen. Ein Dokument ist meistens nur für einen Teil der Benutzer des DMS bestimmt und soll für nicht berechtigte Benutzer nicht zugreifbar sein. Dazu müssen sämtliche Benutzer im System registriert sein und für die Dokumente oder auch einzelne Teile des Dokuments muss definierbar sein, welche Benutzer Aktionen, die mit dem Dokument zu tun haben, ausführen dürfen. Dazu ist es praktikabel, Benutzer in Gruppen einzuteilen, und nach Möglichkeit Zugriffsrechte an Gruppen von Benutzern zu vergeben. Die Gruppen ergeben sich aus der Struktur der Einrichtung, die das DMS einsetzt, etwa den Abteilungen einer Firma. Bei einer Projektmappe sollen beispielsweise nur die Projektmanager den Teil des Angebots einsehen dürfen, während andere Beteiligte nur die Projektbeschreibung sehen sollen.

Benutzer sollten ebenfalls aus Gründen des Datenschutzes nur Informationen über Dokumente erhalten, die sie auch einsehen dürfen, das heißt, wenn ein Benutzer ein Dokument nicht ansehen darf, so soll er auch keine Kenntnis über dessen Existenz erlangen können, da in vielen Fällen die Existenz eines Dokuments oder dessen Bezeichnung schon verwertbare Informationen enthält. Auch innerhalb eines Dokuments muss dieser Grundsatz gelten.

Ein weiterer Punkt hinsichtlich des Datenschutzes sind Gültigkeitszeiträume, die für die Speicherung von Daten gegeben sind oder andersherum, wenn Dokumente erst nach einer bestimmt Zeit zugänglich gemacht werden dürfen.

Aus Gründen der Datensicherheit müssen bestimmte Vorsichtsmaßnahmen beachtet werden. Am wichtigsten ist die Konsistenz der Daten. Die Dokumente oder Teile von Dokumenten müssen bei einer Bearbeitung für andere Bearbeiter gesperrt sein, damit immer nur eine aktuelle Version des Dokuments besteht, es sei denn, eine Aufspaltung ist explizit gewünscht. Wird keine Sperrung durchgeführt, dann lassen sich die Änderungen aus den in Abschnitt 2.2.2.2 genannten Gründen nicht mehr automatisch abgleichen, und keine konsistente Version des Dokuments existiert.

2.3 Anforderungen an Dokumentenmanagement in kooperativen Arbeitsumgebungen

Zunächst soll der Begriff einer kooperativen Umgebung definiert werden, um die Anforderungen, die sich in solchen Umgebungen ergeben, in den folgenden Abschnitten beschreiben zu können. Abgeleitet von der Definition von Groupware aus Schlichter [2, S. 94], kann eine kooperative Arbeitsumgebung als Einsatzgebiet von Groupware wie folgt definiert werden:

> Eine kooperative Arbeitsumgebung bezeichnet ein Arbeitsumfeld, in dem Gruppen mehrerer Personen in Zusammenarbeit an einer gemeinsamen Aufgabe oder für ein gemeinsames Ziel arbeiten.

Typischerweise werden in solchen Arbeitsumgebungen Dokumente erstellt, zu denen jeder beteiligte Mitarbeiter des Unternehmens oder der Behörde einen Teil beiträgt. Diese Umgebungen sind Teil fast jeder Firma, beispielsweise bei der Softwareentwicklung, wo mehrere Programmierer, Graphiker und ein oder mehrere Projektmanager gemeinsam eine Anwendungssoftware erstellen. Eine Vielzahl von Dokumenten entsteht aus Teilleistungen der Mitarbeiter.

In den nächsten Abschnitten werden zuerst die vorhandenen technischen und organisatorischen Infrastrukturen beschrieben (Abschnitt 2.3.1), dann die allgemeinen Anforderungen an ein Dokumentenmanagement in kooperativen Arbeitsumgebungen (Abschnitt 2.3.2) und zuletzt spezielle Anforderungen in Abhängigkeit von der Größe des Unternehmens (Abschnitte 2.3.3 und 2.3.4).

2.3.1 Infrastruktur

Der Aufbau eines DMS in einer IT-Infrastruktur muss selbstverständlich an deren Struktur angepasst sein. In einer Umfrage des Verbundprojektes Adaptive READ [35] [36] [37] zeigte sich, dass die meisten Infrastrukturen heterogen sind, das heißt, es gibt verschiedene Betriebssysteme und unterschiedlich leistungsstarke Computersysteme, auf denen das DMS funktionieren muss. Weiter muss damit gerechnet werden, dass teilweise sehr alte Systeme weiter eingesetzt werden, worauf ebenso alte Betriebsysteme laufen, insbesondere OS/2 oder verschiedene UNIX-Derivate wie etwa AIX.

Es gibt unterschiedliche Herangehensweisen an die Vernetzung der Computer untereinander. Laut Adaptive Read [37] haben 64% der Firmen ein Intranet, und es wird über-

wiegend auf Arbeitplatz-PCs (62%) gearbeitet. Immerhin 27% arbeiten überwiegend mit Workstations und 11% mit Terminals.

Oft gibt es bereits verschiedene Datenbanksysteme, die zur Speicherung von Daten und Dokumenten genutzt werden und für die teure Lizenzen und Sicherungslösungen angeschafft wurden. Diese müssen nach Möglichkeit integriert werden, können keinesfalls aber ersetzt werden, da dies die Kosten für die Einführungen eines DMS erheblich in die Höhe treiben würde.

Ebenso sieht es im Bereich der verwendeten Anwendungssoftware aus. Dazu zählen nicht nur Office-Pakete und noch speziellere Anwendungen, sondern auch der verwendete WWW-Browser oder das E-Mail-Programm. Wie in [37] durch die Umfrage ermittelt wurde, verwenden 61,9% der Firmen Microsoft Outlook, 12,7% Netscape, 4,5% Novell, 1,7% Pegasus-Mail und 19,2% andere E-Mail-Programme. Gerade an den letzteren zeigt sich, dass es nicht möglich ist, sich auf eine bestimmte Software festzulegen und zu spezialisieren.

Aber nicht nur die Infrastruktur der Technik ist für die Planung eines DMS wichtig, auch die Infrastruktur des Unternehmens hinsichtlich des Aufbaus, der Abteilungen und der Verteilung der Mitarbeiter muss berücksichtigt werden. Viele Firmen, insbesondere größere Firmen, sind räumlich weit verteilt mit Niederlassungen in ganz Deutschland oder im Ausland. Ein weiterer Aspekt ist die Telearbeit [38], die laut einer Studie [39] bereits 1999 von europaweit über 5,5 Millionen Arbeitnehmern genutzt wurde, was ca. 4% aller Arbeitnehmer ausmacht. In Deutschland waren es 1.562.000 Arbeitnehmer, die regelmäßig Telearbeit nutzten. Es ist also nicht möglich, davon auszugehen, dass alle Mitarbeiter einer Firma sich auch persönlich sehen, sondern es muss angenommen werden, dass sie mittels Computersystemen und Netzwerken zusammenarbeiten.

Daraus ergeben sich besondere Anforderungen an ein Dokumentenmanagement, die in den folgenden Abschnitten genauer beschrieben werden sollen.

2.3.2 Allgemeine Anforderungen in kooperativen Arbeitsumgebungen

2.3.2.1 Arbeit mit Dokumenten

Die Arbeit mit Dokumenten funktioniert in vielen Firmen noch wie in der Zeit vor der computergestützten Bearbeitung und Verwaltung [37]. So sind viele Dokumente (beispielsweise bei 70% der Firmen die eingehenden Faxe) noch immer in Papierform ab-

gelegt, und nur ungefähr die Hälfte aller befragten Firmen sehen die Überführung von Papierdokumenten in elektronische als wichtige Aufgabe an. Bei der Suche nach Dokumenten werden als wichtigste Suchwerkzeuge bei 14,2% Kollegen, 5,9% Karteikartensysteme und bei 15,3% das Telefon angegeben. Elektronische Systeme kommen dabei zwar auf 64,6%, jedoch sind bei nur 20,9% der Befragten Intranet oder Internet als wichtigstes Recherchesystem genannt worden. Bei der Unterstützung kooperativen Arbeitens darf die typische Arbeitsweise mit Dokumenten keinesfalls außer Acht gelassen werden.

Wichtig ist der Weg eines Dokuments zu seinen Bearbeitern. Dokumente können über Intranets, Datenträger oder E-Mails ausgetauscht oder über Workflowmanagementsysteme automatisch verteilt werden. Wichtig ist aber, wie sich der Fluss der Dokumente verhält, unabhängig vom Transportweg. Es gibt mehrere typische Muster: Dokumente können von einem Bearbeiter an andere Personen zur Korrektur, Ergänzung, Information oder Freigabe weitergegeben werden. Dabei kann es sich bei den nächsten Empfängern um einen oder mehrere Personen handeln. Alle dieser Möglichkeiten müssen in einem Dokumentenmanagement unterstützt werden.

Zeit / Ort	gleichzeitig (synchron)	nicht gleichzeitig (asynchron)	
		vorhersagbar	nicht vorhersagbar
gleicher Ort	face-to-face Sitzung	Schichtarbeit	„schwarzes Brett“
unterschiedliche Orte (vorhersagbar)	Videokonferenz	E-Mail	gemeinsames Verfassen von Dokumenten
unterschiedliche Orte (nicht vorhersagbar)	Mobilfunkkonferenz	Nichtrealzeit-Computerkonferenz	Workflow-management

Tabelle 2.4: Klassifizierung von Groupware nach Zeit und Ort (nach [2, S. 119])

Für die kooperative Arbeit kommen Groupware-Systeme zur Anwendung. Man kann Groupware nach mehreren Kriterien klassifizieren [2, S. 118]. In Tabelle 2.4 sind Groupware-Systeme nach Raum und Zeit klassifiziert. Der Bereich, der das Dokumentenmanagement betrifft, ist die nicht-synchrone Zusammenarbeit an unterschiedlichen Orten. Das liegt an der Tatsache, dass die Personen, die mit Dokumenten arbeiten müssen, jeweils einen eigenen Arbeitsplatz zur Verfügung haben und die Änderungen an den Dokumenten nicht gleichzeitig vornehmen. Für die Unterstützung von gemeinsamem Bearbeiten beispielsweise mit Gruppeneditoren sind je nach Anwendung spezielle Anwendungsprogramme notwendig. Diese Funktionalität kann daher nicht auf alle zu unterstützenden Dokumenttypen verallgemeinert werden.

Groupware kann auch nach der Art der Anwendung klassifiziert werden [2, S. 119–125] [42]. In Tabelle 2.5 sind einige dieser Anwendungen aufgeführt. Das Dokumentenma-

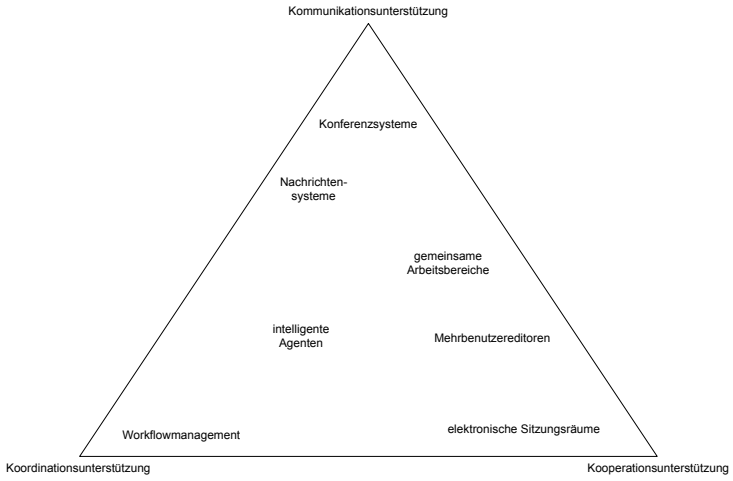

Abbildung 2.3: Klassifikation nach dem 3C-Modell nach [2, S. 125]

nagement betreffen hauptsächlich die gemeinsamen Arbeitsbereiche, die schon viele der typischen Dokumentenmanagement-Funktionalität aufweisen, insbesondere die Funktionen zum Versionsmanagement und gemeinsamen Zugriff auf Dokumente. Aber auch die Funktionen von Nachrichten- und Koordinationssystemen sind im Bereich des Dokumentenmanagement einsetzbar.

Eine weitere Möglichkeit der Klassifikation ist die Einordnung nach dem 3C-Modell [2, S. 125]. Dabei werden die Anwendungsklassen nach der Intensität der Zusammenarbeit in einem Dreieck angeordnet. Es wird zwischen Kommunikations-, Koordinations- und Kooperationsunterstützung unterschieden. Kommunikationsunterstützung gilt als schwächste Stufe und Kooperationsunterstützung als stärkste Stufe der Zusammenarbeit. Wie in Abbildung 2.3 zu sehen ist, werden die Klassen von Groupware-Applikationen so angeordnet, dass sie in der Nähe der von ihnen unterstützten Zusammenarbeitsintensität stehen. Für Dokumentenmanagementsysteme sind alle drei Zusammenarbeitsstufen interessant, womit man sie möglichst in der Mitte des Dreiecks einordnen können sollte.

Als interessante Frage stellt sich, welche Funktionen überhaupt gebraucht werden. An dem BSCW-System wurde untersucht, welche Funktionen die Benutzer von Groupware-Systemen wirklich nutzen [43]. Nur wenige Benutzer verwenden einen Großteil der möglichen Funktionalität, so verwendeten nur 10 von etwa 17.000 Benutzern über 80% der Funktionalität und nur 2.400 Benutzer mehr als 25% der Funktionen. Weiter sind die meisten Zugriffe (72%) nicht modifizierend, was zeigt, dass die Informationsbeschaffung fast dreimal häufiger ist, als alle anderen Operationen. Unter den anderen Features sind

beim regelmäßigen Benutzen die Erstellung (23,1%) und die Modifikation (15,1%) von Daten stark vertreten. Die Awareness-Features, also Funktionen zum Überwachen von Aktivitäten anderer Benutzer, sind mit 17,3% ebenfalls stark genutzt.

Die Überwachung von Dokumenten ist neben den Lesen und Bearbeiten von Dokumenten eine der wichtigsten Funktionen. Alle sinnvollen Ereignisse sollten also überwacht werden können. Dazu gehören Abrufe zum Lesen und Bearbeiten, die Speicherung einer neuen Version, Änderungen an den Metadaten und Benutzerrechten, Abschließen und Löschen des Dokuments und Ablauf von Bearbeitungsfristen. Diese Ereignisüberwachung ermöglicht es dem Benutzer, das Umfeld eines Dokuments besser zu beurteilen und den Weg eines Dokuments besser beobachten zu können. Ohne elektronische Unterstützung erfordert eine Überwachung einen hohen Arbeitsaufwand, der durch die notwendige Kommunikation mit anderen Bearbeitern, die alle beteiligte Mitarbeiter Zeit kostet, entsteht.

Die Kommunikation in Arbeitsgruppen findet mit Dokumenten statt, es ist aber auch notwendig, über die Dokumente zu kommunizieren, diese zu diskutieren. Dazu gibt es mehrere Möglichkeiten [44]. Einerseits bieten viele Anwendungsprogramme (Microsoft Office [45], Adobe Acrobat [46]) Funktionen zum Einfügen von Kommentaren direkt in die Dokumente, andererseits gibt es Diskussionsforen mit Webinterface [47], die wie das Usenet [48] die Möglichkeit zur Diskussion mit Funktionen zur Gruppierung von Diskussionsthemen (Threads) bieten. Solche Kommunikationsmöglichkeiten sind besonders bei räumlich stark verteilten Arbeitsgruppen von Vorteil, da sie asynchron arbeiten und somit unabhängig von Ort und Zeit sind.

2.3.2.2 Abbildung der Wirtschaftsprozesse

In jedem Unternehmen gibt es Wirtschaftsprozesse. In [2, S. 332] wird ein Wirtschaftprozess wie folgt definiert:

> Ein Geschäftsprozess ist eine Ansammlung von sequentiellen und/oder parallelen Aktivitäten, die für die Verarbeitung ökonomisch relevanter Objekte notwenig sind.

Ein Wirtschaftprozess ist also eine Folge von Aktivitäten mit klar definiertem Ablauf, die wiederholt auftritt. Die Wirtschaftprozesse in einem Unternehmen beinhalten die Erstellung von Dokumenten. Also muss ein DMS den durch die Wirtschaftsprozesse gegebenen Anforderungen gerecht werden und die Prozessabarbeitung unterstützen. Da an Wirt-

schaftsprozessen üblicherweise mehrere Personen beteiligt sind, wird an den zugehörigen Dokumenten gemeinsam gearbeitet. Dazu durchlaufen die Dokumente das Unternehmen von Mitarbeiter zu Mitarbeiter.

Typischerweise fließen die Dokumente innerhalb eines Unternehmens durch mehrere Hierarchieebenen. Je nach Größe des Unternehmens oder der Institution sind die Hierarchien sehr flach oder – meist mit wachsender Mitarbeiterzahl – sehr stark strukturiert. Diese Spanne muss mit einem DMS gut verträglich sein, wenn der Dokumentenfluss nicht ins Stocken geraten soll. Einerseits sollen Strukturen zur Entscheidung, wie etwa die Projektleiter/Projektmitarbeiter oder die Sekretärin, die Dokumente vorbereitet bzw. der Manager, der über die Freigabe entscheidet, unterstützt werden. Andererseits darf das DMS nicht unpraktisch in flachen Hierarchien sein, in denen die Rollen der Mitarbeiter oft wechseln können.

Generell ist es wünschenswert, dass die aus dem nichtelektronischen Arbeiten entstandenen Arbeitsweisen mit Dokumenten adaptiert werden können. Solche Mechanismen sind auch in der Arbeit mit elektronischen Dokumenten üblich, wie beispielsweise das Versenden von Dokumenten per E-Mail.

In mittleren und großen Unternehmen werden zur Unterstützung und Ausführung von Geschäftsprozessen häufig Workflowmanagementsysteme eingesetzt [49]. Diese Systeme übernehmen die Koordination der einzelnen Schritte und Abläufe eines Wirtschaftsprozesses und der damit verbundenen Daten. Mittels komplexer Beschreibungssprachen können Arbeitsabläufe modelliert und spezifiziert werden und dann abgearbeitet, kontrolliert und analysiert werden. Es gibt unterschiedliche Herangehensweisen von Workflowmanagementsystemen [50]. Im Gegensatz zu den prozessorientierten Systemen kommen die dokumentorientierten Systeme (zum Beispiel zum Document Routing) Dokumentenmanagementsystemen sehr nahe, da sie die zu bearbeitenden Dokumente mit den Flussinformationen versehen und von Bearbeiter zu Bearbeiter transportieren. Vereinfachte Funktionen solcher Systeme sollten auch in Dokumentenmanagementsystemen angeboten werden, die kooperative Arbeit unterstützen. Ein DMS sollte weiter die volle Funktionalität eines dokumentorientierten Workflowmanagementsystem durch Schnittstellen oder modulare Erweiterungen erreichen können.

2.3.2.3 Adhoc-Workflows

Die Ausführung der Wirtschaftsprozesse ist ständigen Veränderungen unterworfen. Das Workflowmanagement muss auf diese Änderungen reagieren und sie in die modellierten

Workflows übernehmen. Je nach der Häufigkeit der Änderungen kann dieses Vorgehen sehr aufwändig sein. Gerade in kleinen Firmen ist dieses Problem vorhanden, da diese sehr flexibel arbeiten müssen und die Kosten für das Reengineering nicht aufgebracht werden können. Aus diesen Gründen unterscheidet man zwischen Production Workflows, die bis ins Detail definiert werden können und wenige dynamische Informationsflüsse aufweisen, und Adhoc-Workflows, die keine feste Struktur aufweisen und stark fallbezogen sind [2, S. 336]. Adhoc-Workflows können nicht immer im Voraus definiert werden oder müssen entsprechend allgemein gefasst werden können. Ein weiteres Problem ist die Tatsache, dass Workflows oftmals spontan angepasst werden müssen. Die Modellierung der Workflows ist aber zu kompliziert, um von allen Benutzern des Systems durchgeführt zu werden.

Zur Lösung muss es im DMS möglich sein, einfache Workflows spontan zu definieren. Beispielsweise entscheidet der Projektleiter bei der Softwareentwicklung, welcher Mitarbeiter ein Softwaremodul ändern soll und schickt ihm eine Aufgabenbeschreibung und das entsprechende Modul mit genauen Anweisungen, bis wann die Arbeit abgeschlossen sein soll. Gerade in kleinen Gruppen und flachen Hierarchien ist diese Vorgehensweise viel praktikabler, als die Definition eines neuen Workflows. Zu den Funktionen, die notwendig sind, um diese Adhoc-Workflows in einem DMS zu realisieren, gehören die Möglichkeit, Dokumente mit Aufgabenbeschreibungen zu versenden, Bearbeitungsfristen zu setzen, mehrere mögliche Bearbeiter zu definieren bzw. die Aneinanderreihung solcher Elemente.

Workflowmanagementsysteme ermöglichen ebenfalls die Überwachung der Fortschritte innerhalb der modellierten Geschäftsprozesse. Bei der oben vorgeschlagenen Lösung geschieht das anhand von Arbeitslisten mit einer Auflistung der offenen Aufgaben zu den Bearbeitern und die Möglichkeit, Ereignisse bezüglich der Dokumente zu überwachen.

2.3.3 Spezielle Anforderungen in kleinen Umgebungen

Die Zielgruppe für das zu konzeptionierende DMS soll möglichst groß sein. Also sollen auch Arbeitsumgebungen unterstützt werden, die sonst aus Kosten- oder Praktikabilitätsgründen auf den Einsatz eines DMS verzichten würden. Auch in kleinen Umgebungen kann der Einsatz von CSCW-Systemen sinnvoll sein [51], speziell zur Förderung der Koordination und Awareness. Dazu gehören insbesondere kleine Firmen, die die doch erheblichen Anschaffungskosten für am Markt erhältliche Dokumentenmanagementsysteme scheuen. Eine genaue Eingrenzung ist schwer möglich, als Richtwert soll aber eine

Mitarbeiterzahl von maximal 50 mit dem Computer arbeitender Personen für kleine kooperative Umgebungen angenommen werden.

Es spielen zwei grundlegende Faktoren eine Rolle, wenn es um die Verwendung eines DMS geht. Einer dieser Faktoren ist der mit dem Einsatz verbundene Overhead, das heißt die Menge der zusätzlichen Arbeit, die durch den Einsatz entsteht. Dieser Overhead muss durch die Zeitersparnis, die durch das DMS erreicht wird, aufgehoben werden. Dies erfordert eine möglichst einfache Bedienung und Konfiguration des DMS. Hinsichtlich der Konfiguration bedeutet das, dass beispielsweise kein Mitarbeiter sich größtenteils mit der Pflege des DMS beschäftigen muss. Auf das Thema Bedienbarkeit, also die Usability, wird weiter unten in Abschnitt 2.5 genauer eingegangen.

Der zweite Faktor sind die Kosten für die Einführung des DMS selbst. In kleinen Umgebungen werden oft keine geeigneten Datenbanksysteme eingesetzt, die als Datenspeicher in Frage kommen würden, es sind selten Applikationsserver vorhanden, es ist nicht zwingenderweise ein leistungsstarker Server vorhanden. Diese Infrastruktur kann daher nicht vorausgesetzt werden. Wenn aber zum Einsatz des DMS weitere teilweise sehr kostspielige Lizensierungen für Software und Investitionen für Hardware notwendig sind, wird darauf verzichtet. Dem zufolge muss das DMS auch in einer Version mit diesen Schwierigkeiten umgehen können. Ein weiterer Punkt, der die Kosten eines DMS in die Höhe treibt, sind Funktionen, die in vielen Fällen nicht notwendig sind. So braucht nicht jede Firma ein Workflowmanagement oder eine Volltextindizierung. Das DMS muss also modular aufgebaut werden und für alle Funktionen eine einfache, kostengünstige und eine qualitativ hochwertige Variante haben.

2.3.4 Spezielle Anforderungen in mittleren und großen Umgebungen

Als mittlere und große Umgebungen sollen im Rahmen dieser Diplomarbeit Arbeitsumgebungen mit mehr als 50 Mitarbeitern beziehungsweise 500 Mitarbeitern gelten. Ebenso wie für kleine Umgebungen kann man diese Grenzen als Schätzwert ansetzen, die unter Umständen noch stark variieren können.

Charakteristisch für diese Umgebungen ist eine sehr heterogene Struktur. Die Mitarbeiter sind in Abteilungen aufgeteilt, und die Bildung von Hierarchien ist umso stärker ausgeprägt, je mehr Mitarbeiter die Firma hat. Die Zusammenarbeit findet zunächst innerhalb der Abteilungen statt, so dass ein Großteil der Dokumente nur in der eigenen Abteilung verwendet wird. Eine Eingrenzung ist aber nicht sinnvoll, da es oft vorkommt, dass einzelne Arbeitsgruppen abteilungsübergreifend gebildet werden, Dokumente aus

anderen Abteilungen verwendet werden oder mehrere Abteilungen zusammenarbeiten. Mit dem DMS muss es möglich sein, alle in der Umgebung vorkommenden Strukturen abzubilden. Es darf deshalb keine feste Struktur vorgegeben sein, die zum Beispiel die Mitarbeit einer Person in mehreren Arbeitsgruppen, auch unterschiedlicher Abteilungen, verhindert.

Weiter gilt es zu beachten, dass die Mitarbeiter hier stärker auf ihre Arbeitsaufgaben spezialisiert sind. Man kann nicht bei jedem Mitarbeiter von demselben technischen und organisatorischen Verständnis ausgehen, wie das in kleineren Umgebungen noch möglich ist. Aus diesem Grund muss das DMS Möglichkeiten zur Einschränkung der Funktionalität der einzelnen Benutzer geben. Eine Sekretärin benötigt nicht dieselben Funktionen wie ein Abteilungsleiter oder gar ein Geschäftsführer. Teilweise gibt es auch rechtliche Bestimmungen, die eine Einschränkung fordern, oder es ist aus Gründen der Datensicherheit geboten, solche Einschränkungen beim Zugriff zu machen.

Aus diesem Grund muss das DMS sehr fein granuliert konfigurierbar sein und auf Benutzerebene umfangreiche Einschränkungen erlauben. Dies ist aber, wenn man die im vorhergehenden Kapitel geforderte einfache Konfigurierbarkeit einbezieht, eine Herausforderung an das Design der Software. Es gilt hier sicherlich, den modularen Aufbau weiter zu nutzen, um diese Zielstellung zu erfüllen.

Je größer die Firma wird, desto mehr Funktionen werden sinnvoll eingesetzt. Dabei steht die Kostenfrage nicht so sehr im Vordergrund wie in kleinen Unternehmen, was unter anderem auch die Vielzahl von Anbietern im Bereich des Enterprise Document Manegement, also dem Dokumentenmanagement für große Wirtschaftsunternehmen, zeigt. Hier müssen dann individuell angepasste Lösungen mit einer Vielzahl von Optionen wie Volltextindexierung der Dokumente, Workflowmanagement oder die Anbindung an andere bereits vorhandene Dienste geschaffen werden können. Das schließt nicht zuletzt die Integration von Datenbanksystemen ein, die bereits lizensiert und gut mit Backuplösungen ausgestattet sind.

Ein weiteres Problem bei großen Umgebungen ist naturgemäß die Leistungsfähigkeit der Software. Mit der Anzahl der Nutzer wächst die Anforderung an die Hardware und Software. Das DMS muss mit großen Dokumentmengen und gleichzeitigen Benutzern immer noch performant arbeiten und darf keine langen Wartezeiten beim Benutzer erzeugen.

2.4 Notwendige Schnittstellen

In den vorangegangenen Abschnitten wurde die notwendige Funktionalität eher allgemein beschrieben. In diesem Abschnitt sollen die notwendigen Schnittstellen zu anderen Systemkomponenten und dem Benutzer noch einmal genau benannt werden.

2.4.1 Benutzerverwaltung

In Intranets wie in jedem anderen Firmennetzwerk müssen sich Benutzer authentifizieren. Dazu erhalten sie Konten, denen sie durch geeignete Mechanismen wie Passwörter oder Chipkarten mit komplexeren kryptographischen Verfahren zugeordnet werden, sich also authentifizieren. Ist dies erfolgreich durchgeführt worden, dann kann der Benutzer im Netzwerk arbeiten.

Normalerweise wird ein solches Authentifizierungssystem bereits eingesetzt. Deshalb ist es wünschenswert und oft auch notwendig, dass dieses System zur Authentifizierung vom eingesetzten DMS verwendet wird. So müssen die Benutzer sich nicht mehrere Passwörter merken, was auch die Akzeptanz erhöht.

Die Authentifizierungssysteme sind üblicherweise im Betriebssystem integriert, so dass das Benutzerkonto des Betriebssystems direkt auf das Benutzerkonto im DMS abgebildet werden könnte. Insbesondere bei zentraler Verwaltung der Konten auf Netzwerkservern ist dies mit relativ wenig Aufwand möglich.

Oft sind Verzeichnisdienste wie LDAP [52] zum Zugriff auf diese Kontenverwaltung im Einsatz, um einerseits die für die Authentifizierung notwendigen Daten zu speichern und andererseits auch Daten zum Benutzer selbst zu speichern. Dazu gehören unter anderem die Berechtigungen des Benutzer für bestimmte Aktionen, wie zum Beispiel Ausführungsrechte zu Programmen. Diese Verzeichnisdienste sind meistens mit Datenbanken und entsprechenden Backupsystemen gekoppelt, so dass es sich anbietet, sie ebenfalls für die Speicherung der DMS-spezifischen Daten zu verwenden. Dies sollte wiederum optional sein, da sie nicht immer im Einsatz und wieder mit technischem und finanziellem Aufwand verbunden sind.

2.4.2 Suche im Dokumentenbestand

Die Suche im Dokumentenbestand ist eine der wichtigsten Aktionen in einem DMS. Dokumente müssen schnell gefunden werden können. Das erfordert präzise Ergebnisse bei der Recherche im Dokumentenbestand. Laut der Umfrage von Adaptive Read [37] sehen 60,4% in der Informationssuche eine wichtige Aufgabe, jedoch verwenden nur 15,5% dazu ein elektronisches Recherchesystem und 20,9% das Intranet. Die übrigen Anteile bestehen aus 35,5% für nicht-elektronische Suchwerkzeuge wie Karteien oder Kollegen und 28,1% für Dateisysteme als primäres Recherchewerkzeug. Immerhin verwenden 55% der Befragten mehr als 10% ihrer täglichen Arbeitszeit für Informationssuche, davon 49,8% zwischen 10% und 30%.

Für gutes Dokumentenmanagement ist es daher unabdingbar, ein zuverlässiges Werkzeug zur Rechereche im Dokumentenbestand anzubieten, sei es im Intranet oder als gesondertes Werkzeug. Auch eine Anbindung an bereits vorhandene Systeme sollte möglich sein.

2.4.3 Unterstützung von Workflows

In Abschnitt 2.3.2.2 wurde bereits auf die Gemeinsamkeiten zwischen Workflowmanagement- und Dokumentenmanagementsystemen eingegangen. Bereits mit den in Abschnitt 2.3.2.3 vorgeschlagenen Funktionen für die selbständige Definition von Workflows kann eine breite Palette an Workflowfunktionen abgedeckt werden. Daher ist eine direkte Integration von Workflowmanagementfunktionalität nicht notwendig. Es sollte aber trotzdem möglich sein, sowohl eine eigene Workflowkomponente als auch Workflowmanagementsysteme von Fremdherstellern in das Dokumentenmanagementsystem integrieren zu können. Alternativ würde ebenfalls die Möglichkeit bleiben, ein Workflowmanagement über das Dokumentenmanagement zu stellen, so dass nur Verweise auf die Dokumente den Workflows beigefügt werden können.

Diese Anforderungen ergeben sich aus der Tatsache, dass am Markt bereits sehr viele Workflowmanagementsysteme vorhanden sind und oft schon in Unternehmen eingesetzt werden (bei immerhin 13,2% der von Adaptive Read [37] befragten), die in der Zielgruppe des neu zu entwickelnden DMS liegen.

2.4.4 Datenaustausch mit dem Benutzer

Eine der zentralen Fragen des Dokumentenmanagement ist, wie die Dokumente zum Benutzer kommen, und, wie sie vom Benutzer zum DMS kommen. Die Lösung sollte so einfach wie möglich sein, sowohl in der Bedienung als auch in der Einbindung in DMS und Betriebssystem.

Für den Dateiaustausch mit Fremdrechnern gibt es sehr viele unterschiedliche Systeme. Einmal gibt es Netzwerkdateisysteme wie das unter UNIX-Derivaten verbreitete NFS [53] oder das in allen aktuellen Windows-Versionen verwendete SMB-Protokoll [54]. In diesen Systemen wird die Verbindung zum entfernten Dateisystem automatisch aufgebaut und geschieht unsichtbar für den Anwender. Die Dateisysteme erscheinen in der Verzeichnisstruktur des UNIX-artigen Betriebssytems oder als Laufwerk bei den Windowsbetriebssystemen von Microsoft und unterscheiden sich in der Verwendung nicht von lokalen Dateisystemen.

Für die Übertragung von Dateien von entfernten Rechnersystemen auf lokale Systeme gibt es das Internetprotokoll FTP [55] oder Dienste wie SSH [56] mit Programmen wie scp (secure copy). Clientanwendungen existieren für nahezu alle Betriebssysteme [57]. Hier muss der Anwender explizit eine Verbindung aufbauen, die durch eine Passwortabfrage authentifiziert und autorisiert wird. Danach kann das Client-Programm auf die Verzeichnisstruktur zugreifen und Daten zwischen den Computern austauschen. Auch Applikationen wie CVS [15] verwenden zum Zugriff auf den Dateibestand auf entfernten Rechnersystemen ein eigenes Client-Programm. Dazu muss ebenfalls eine Verbindung vom Benutzer explizit aufgebaut werden.

Ein weiteres Protokoll zum Dateitransfer in Netzwerken ist World Wide Web Distributed Authoring and Versioning (WEBDAV) [58] [59]. Die Internet Engineering Taskforce (IETF) hat WEBDAV [60] als Erweiterung des Hypertext Transfer Protocol (HTTP) [6] zur Unterstützung des verteilten Erstellens von Dokumenten verabschiedet. Dazu wurde der HTTP-Standard um Funktionen zum Speichern, Sperren, Löschen von Dateien, Verwalten von Verzeichnisstrukturen (Collections) und Speichern von Metadaten erweitert. Das Protokoll findet eine gute Untertützung von Softwareherstellern. So ist es bereits in die aktuellen Versionen von Windows unter dem Namen „Webordner“ integiert. Auch für andere Betriebssysteme existieren Clientanwendungen [61].

Es gibt einige Erweiterungen zum WebDAV-Standard, etwa DeltaV, die Versioning Extensions [62] [63] zur Versionierung der Dokumente. WebDAV befindet sich hinsichtlich

wesentlicher Aspekte jedoch noch in Entwicklung. Dazu gehören das Management von Benutzerberechtigungen [64], Sortierung von Collection-Inhalten [65] und Umleitungen auf andere Server [66]. Aber gerade im Bereich des CSCW und des Dokumentenmanagements hat WebDAV schon zur Entwicklung neuer Software und zu Forschungsarbeit geführt [67] [68] [69].

2.4.5 Speicherung der Daten

Das Dokumentenmanagementsystem ist dafür verantworlich, dass die Dokumente sicher und konsistent gespeichert werden. Dazu ist es günstig, die Daten zentral auf einem Server abzuspeichern. Dies hat den Vorteil, dass für die Erstellung von Sicherungskopien nur ein zentraler Punkt angelaufen werden muss. Ein weiterer Vorteil ist, dass keine Abgleiche mit lokalen Datenbeständen bei Änderungen an einem Dokument durchgeführt werden müssen.

Die Speicherung selbst sollte aber von einer anderen Systemkomponente als dem DMS durchgeführt werden, da es bereits mehrere ausgereifte Lösungen dafür gibt. Eine der gängigsten Möglichkeiten ist die Speicherung in einem Datenbankmanagementsystem. Datenbanken haben den Vorteil, dass sie durch Transaktionen geschützte persistente Speicherung über Programmierschnittstellen ermöglichen.

In vielen IT-Infrastrukturen sind bereits Datenbanksysteme im Einsatz. Daher ist es sinnvoll, dass das DMS auf diese Systeme zurückgreift, um die eigenen Daten zu speichern. Da die Programmierschnittstellen zu Datenbanken sehr verschieden sind, stellt die Unterstützung von möglichst vielen Systemen eine Herausforderung dar, inbesondere deshalb, weil der Funktionsumfang der Systeme stark variiert.

Sollte eine Datenbank nicht vorhanden sein, dann kann auch auf ein Dateisystem auf dem Server-Computer des DMS zurückgegriffen werden. Dies wäre eine kostensparende Alternative zu den hohen Lizensierungskosten von Datenbanksystemen. Dann muss aber mehr Aufwand bezüglich Persistenz und Konsistenz des Datenbestandes im Dateisystem betrieben werden, da diese Funktionen nicht von dem für das Dateisystem verantwortliche Betriebssystem unterstützt werden. Daher muss der Einsatz einer Datenbank den Vorrang vor einer solchen Lösung haben.

Weiter muss das DMS die Erstellung von Sicherungskopien unterstützen, um inkrementelle Sicherungen zu ermöglichen, die gerade bei einem sehr großen Datenbestand wichtig sind. Eine eigene Backuplösung selbst muss das DMS hingegen nicht unterstützen, da

es sowohl für Datenbanken als auch für Dateisysteme genügend Lösungen zu Sicherung des Datenbestandes gibt.

2.5 Akzeptanzkriterien für die Einführung von Dokumentenmanagementsystemen

Viele Firmen setzen kein Dokumentenmanagementsystem ein (laut Adaptive Read [37] 70,8%), planen aber den Einsatz in der Zukunft. Es gibt neben dem Kostenfaktor und der Grundfunktionalität, die oben erläutert wurde, noch weitere Kriterien, die erfüllt sein müssen, damit ein DMS Akzeptanz findet.

Als Kriterien für die Systemauswahl kommen auch nutzungsorientierte Kriterien zum Zuge [70]. Dazu zählen Benutzbarkeit, Gebrauchstauglichkeit und Integrierbarkeit. Normalerweise wird Systemen, die sich gut in den bisherigen Arbeitsablauf einpassen lassen und die einfach zu bedienen sind, der Zuschlag gegeben. Es muss also in besonderem Maße auf die Usability geachtet werden. Systeme, die nicht verstanden werden, werden im Zweifelsfall nicht mehr verwendet und enden als Fehlinvestition [71], was schlecht für das Ansehen der Software im Markt und damit auch schlecht für die Verkaufszahlen ist. Die Usability beeinflusst den Aufwand, der zum Betrieb des DMS notwendig ist und so klein wie möglich sein soll. Das schließt unter anderem Administrierung des Systems mit ein.

Das DMS muss ebenfalls in die Infrastruktur des Unternehmens passen. Mangelnde Unterstützung für bestimmte Betriebssysteme kann schnell zum K.O.-Kriterium werden, wenn Firmen kein Windows einsetzen (siehe Abschnitt 2.3.1).

Viele Systeme haben wesentlich mehr Funktionen, als überhaupt verwendet werden. Eine Überforderung des Benutzers mit zu vielen Funktionen kann abschreckend wirken und sollte vermieden werden. Auch kann das Produkt als zu umfangreich eingeschätzt werden und aus diesem Grund nicht zum Einsatz kommen. Daher ist es üblich, die Softwaresysteme modular zu konzipieren, so dass die benötigten Komponenten bei Bedarf zugekauft werden können (siehe Abschnitt 2.6).

2.6 Anforderungsevaluierung bei vorhandenen Dokumentenmanagementsystemen

Es befinden sich viele verschiedene Systeme auf dem Markt. Die für die in den vorhergehenden Abschnitten genannten Anforderungen in Frage kommenden Systeme werden von den Herstellern oft direkt vermarktet. Es hat sich als schwierig herausgestellt, detaillierte Informationen zu diesen Systemen zu bekommen, da das Material, dass auf den Websites der Anbieter angeboten wird, nur wenige Informationen zum Funktionsumfang bietet. Nur in seltenen Fällen gibt es eine Demonstrationsversion, die den Umfang des Systems veranschaulicht. Auch auf Anfragen erhält man fast ausschließlich Verweise auf das Präsentationsmaterial und keine genauen Informationen. Daher erhebt die folgende Übersicht keinen Anspruch auf Vollständigkeit und gibt hauptsächlich aus Werbematerial gewonnene Informationen wieder.

Ebenso gilt es zu beachten, dass die meisten Systeme nicht ausschließlich zum Dokumentenmanagement konzipiert sind, sondern ebenso Content-, Workflow- oder Knowledgemanagement als wesentlichen Bestandteil des Systems bieten [72] [73] [74] oder als Erweiterungen zu Datenbanklösungen angelegt sind [75].

In den Tabellen 2.6 und 2.7 sind verschiedene Dokumentenmanagementsysteme aufgeführt. Als problematisch kann erachtet werden, dass viele Systeme nur für eine eingeschränkte Zahl von Betriebssystemen verfügbar sind, hauptsächlich Windows. Auch die Möglichkeit, eine kostengünstige Datenbank oder das Dateisystem zur Speicherung der Daten zu nutzen, ist nur in wenigen Fällen gegeben. Insbesondere im unteren Preissegment gibt es nur DocuShare von Xerox [76] mit Unterstützung für kooperative Arbeitsumgebungen. Ein weiteres Problem ist, dass das Konzept der Verbunddokumente nur sehr selten verwendet wird bzw. sich oft nur emulieren lässt.

2.7 Gründe für die Notwendigkeit einer Neuentwicklung

Bevor ein Softwaresystem neu entwickelt wird, muss geklärt werden, ob überhaupt Bedarf an einem System besteht. Nur dann lohnt sich der Aufwand einer Entwicklung.

Es gibt mehrere Gründe, die für gute Chancen einer Neuentwicklung sprechen:

1. Die Anzahl der Systeme, die für kleine Umgebungen geeignet sind, ist gering, insbesondere dann, wenn man Unterstützung kooperativer Arbeit fordert.

2. Nur wenige Systeme bieten bei der Wahl von Server und Clients genügend Möglichkeiten, was die Integration von vielen verschiedenen Betriebssystemen angeht. Gerade Linux und MacOS werden noch nicht ausreichend unterstützt.

3. Viele Systeme können nur teure Datenbanksysteme verwenden und bieten keine Möglichkeit, auch das Dateisystem des Servers oder Datenbanken mit Open-Source-Lizenzmodellen wie PostgreSQL [85] einzubinden.

4. Die Unterstützung von Verbunddokumenten zur besseren Strukturierung der Dokumente und Erleichterung der Arbeit mit zusammengehörigen Dateien ist selbst in vielen Produkten der gehobenen Preisklasse kaum implementiert.

Das genaue Ziel muss also sein, ein DMS für kleine und mittlere Umgebungen zu entwickeln, dass die kooperative Arbeit unterstützt und modular sehr flexibel an- und eingepasst werden kann. Der Markt für große Umgebungen hingegen ist bereits sehr gut abgedeckt.

Anwendung	Beschreibung
Nachrichtensysteme	dienen zum Austausch von Nachrichten zwischen den Benutzern des Systems und können zum Weitergeben von Dokumenten benutzt werden. Beispiel: E-Mail.
Mehrbenutzereditoren	erlauben das gemeinsame Bearbeiten eines Dokuments nach dem WYSIWIS-Prinzip (what you see is what I see) und sind beispielsweise bei der Entwicklung von Software nützlich. Mehrbenutzereditoren gibt es sowohl in synchroner als auch asynchroner Ausprägung. Die Benutzer des Editors werden gegenseitig über die Aktivitäten der anderen informiert und die bearbeiteten Bereiche im Dokument werden isoliert, um Konflikte zu verhindern.
Elektronische Sitzungsräume	werden in Group-Support-Systemen eingesetzt, um die Entscheidungsfindung zu unterstützen. Weitere Anwendungen sind Problemanalyse und -diskussion, Brainstorming oder Abstimmungsdurchführung und -auswertung.
Konferenzsysteme	ermöglichen die Interaktion zwischen räumlich und/oder zeitlich verteilten Personen. Beispielanwendungen sind Video- und Audiokonferenzsysteme, aber auch Chat-Systeme.
Intelligente Agenten	sind Programme, die Aufgaben von Personen übernehmen können und selbständig in Konferenzsystemen arbeiten. Eine Anwendung ist zum Beispiel die Protokollierung von Sitzungen.
Koordinationssysteme	werden eingesetzt, um die Koordination der Arbeit der Gruppenmitglieder zu verbessern. Diese Arbeit ist meist asynchron. Es gibt unter anderem formularbasierte, konversations- und kommunikationsorientierte Systeme. Diese Systeme können auch unter Einbindung anderer Anwendungsklassen arbeiten, etwa wenn Nachrichten- oder Konferenzsysteme zur Koordination der Gruppenaktivitäten eingesetzt werden.
Gemeinsame Arbeitsbereiche	sind Systeme, in denen die Kommunikation innerhalb der Gruppe über gemeinsam zugreifbare Dokumente abgewickelt wird. Ziel ist das gemeinsame Erstellen von Dokumenten. Die Funktionen gemeinsamer Arbeitsbereiche (shared information workspaces) umfassen die persistente Speicherung und entsprechende Mechanismen zum Zugriff auf Dokumente. Es gibt unterschiedliche Modi der Zugriffsteuerung: • *Einzelverantwortung:* Die Benutzer sind jeweils für einen seperaten Teil der Dokumente verantwortlich. Pro Dokument gibt es nur einen Bearbeiter. • *Wechselseitig ausschließender Zugriff:* Alle Teammitglieder können auf ein Dokument zugreifen. Für die Bearbeitungszeit muss das Dokument für weitere Bearbeiter gesperrt werden. • *Alternative Versionen:* Jedes Teammitglied arbeitet an einer eigenen Version des jeweiligen Dokuments. Um eine konsistente Fassung zu erhalten, müssen die einzelnen Dokumente zusammengefasst werden. Dies kann jedoch nicht immer automatisch geschehen. Diese Problematik wurde bereits in Abschnitt 2.2.2.2 diskutiert. • *Synchroner Zugriff:* Der Zugriff auf die Dokumente erfolgt mit Gruppeneditoren. Beispiele für diese Art von Systemen sind Lotus Notes [40] oder BSCW [41].

Tabelle 2.5: Klassifizierung von Groupware nach Art der Anwendung (nach [2, S. 119–125] und [42])

Produkt	Beschreibung
Documentum [77]	Großer Anbieter von Content- und Dokumentmanagementprodukten. System unterstützt Windows NT/2000, Solaris 7/8, AIX 4.3, HPUX als Betriebssystem und Oracle 8, Sybase 11, Microsoft SQL Server 7/2000 und IBM DB2 als Datenbank. Es arbeitet mit Browser-Plugins für Internet Explorer und Netscape. Die Technologie basiert auf Java 2 Enterprise Edition mit breiter Unterstützung von XML. Der Preis des kompletten Systems (inkl. Content Managemet) liegt zwischen 140.000 und 720.000 Euro.
Xerox DocuShare [76]	Sehr umfangreiches DMS-Produkt mit Groupware-typischen Funktionen wie Document Routing und Content Management. Unterstützt Windows, Solaris und Linux als Server und beliebige Platformen als Client (Browserzugriff). Verwendet WebDAV; MSDE Relational Database / PostgreSQL als Datenbank, optional auch Oracle, SQL Server. Preis ab 4.300 Euro, Preis der Lizensierung nach Anzahl Prozessoren/Server und Clients.
FileNet [73]	Kombinierte Content- und Dokumentmanagement-Lösung. Benötigt Windows auf Clientseite; Windows NT, HPUX 11i oder Solaris 8 als Server möglich. MS SQL Server oder Oracle 8 als Datenbank. Unterstützt WebDAV und verteilte Server. Basiert auf Java 2 Enterprise Edition. Preis ab 110.000 Euro.
Hummingbird [78]	Bietet breite Palette von Content-, Workflow- und Dokumentmanagement-Produkten mit Untertützung für kooperatives Arbeiten. Benötigt Windows NT/2000 als Server und Windows als Client. Oracle, MS SQL Server und Sybase sind als Datenbanken möglich. Preis ab 4300 Euro für DMS-Teil.
Gauss VIP DocManager [72]	Deutscher Anbieter mit breiter Palette für Content- und Workflowmanagement. Unterstützt alle wichtigen Betriebssysteme als Server und Client; Oracle 8i, IBM DB2, MS SQL Server als Datenbank. Basiert auf Java 2 Enterprise Edition. Preis ab ca. 38.000 Euro.

Tabelle 2.6: Marktübersicht über Dokumentenmanagementsysteme (1/2)

Produkt	Beschreibung
Hyperwave [79]	Hersteller für Content- und Dokumentenmanagement. Unterstützt Windows, Linux, HPUX, AIX und Solaris als Server/Client; MS SQL Server und Oracle. Verwendet WebDAV.
Infonea [74]	Produkt zum Enterprise Knowledge Management mit integriertem DMS. Server benötigt Windows NT/2000, Client benötig Webbrowser. Als Datenbank ist Oracle oder MS SQL Server möglich.
IXOS eCONserver [80]	DMS mit Erweiterungsmöglichkeiten zu Workflow- und Contenmanagement sowie Groupware. Integration in Microsoft Exchange, SAP- und andere Systeme als primäre Strategie. Als Server kommen UNIX und Windows 2000 zum Einsatz. Clients für Windows bzw. über Integration. Oracle oder MS SQL Server 2000 als Datenbank.
LiveLink [81]	DMS mit umfangreichen Knowledge- und Contentmanagementfunktionen. Integration in SAP und Lotus Notes möglich. Verwendet CORBA und XML. Server für Windows, Solaris und HPUX; Datenbanken Oracle, Sybase und MS SQL Server. Client nur mit Windows praktikabel. Preis ab ca. 70.000 Euro.
Lotus Domino.Doc [82]	System basiert auf Lotus Domino. Benötigt Lotus Domino Server/Windows NT/2000, Lotus Notes Client für Windows. Preis ab ca. 7.000 Euro.
Oracle Files [75]	Erweiterung zur Oracle 9 Datenbank. Dokumente werden in der Datenbank verwaltet. Für fast alle Plattformen erhältlich. Unterstützt WebDAV.
Saperion [83]	Lösung für Dokumentenmanagement mit Möglichkeit zur Anbindung von Workflowmanagement. Unterstützt Windows, Linux und Solaris Client- und Server-seitig. Datenbankanbindung über ODBC. Unterstützung für kleine Umgebungen. Preis ab 5000 Euro (für 10 Benutzer).
Docuware [84]	Deutsches Produkt mir Zielgruppe Mittelstand (5-5000 Benutzer). Volle Integration in Windows, keine anderen Betriebssysteme. Datenbankanbindung mittels ODBC, Dokumente nicht in der Datenbank. Preis ab 3500 Euro.

Tabelle 2.7: Marktübersicht über Dokumentenmanagementsysteme (2/2)

3 Entwurf und Design

In diesem Kapitel wird das entworfene System beschrieben und dessen Funktionsweise detailliert erläutert. Zunächst soll der grundsätzliche Aufbau beschrieben werden (Abschnitt 3.1). Das umfasst sowohl die Sicht des Benutzers als auch die interne Struktur der Software. Danach wird auf die verwendeten Technologien eingegangen (Abschnitt 3.2) und darauf, wie die Entscheidungen zustande gekommen sind. Im Anschluss werden die Details der Architektur des Systems beschrieben (Abschnitt 3.3) und die Funktionsweise erklärt.

Alle Entscheidungen sollen anhand von Kapitel 2 begründet werden.

Es wird jedoch aus Gründen des Umfangs nur auf das Kernsystem und das Modul zur Untertützung kooperativer Arbeit umfassend eingegangen. Die Teile für Retrieval und Workflowverwaltung sollen nur grob diskutiert werden.

3.1 Struktur des Softwaresystems

Das Dokumentenmanagementsystem ist eine Client/Server-Applikation. Dieses Konzept hat mehrere Vorteile. Einmal kann das System zentral auf einem leistungsfähigem Computer installiert werden und die Anwender benötigen keinen leistungsstarken Rechner. Weiter ist die Pflege und Administrierung der zentralen Komponente der Software nur auf einem Computer notwendig, was den notwendigen Arbeitsaufwand absenkt.

In Abbildung 3.1 wird die Architektur schematisch dargestellt. Links der gestrichelten Linie befindet sich die Client- und rechts davon die Serverseite des Systems. Die Serverseite des Dokumentenmanagementsystems besteht aus drei Schichten. Die oberste Schicht besteht aus dem Web-Client, der mit der zweiten Schicht, dem Kernsystem, kommuniziert. Die Speicherung der Daten selbst erfolgt wieder über eine weitere Schicht, den Datenspeicher. Damit sind Präsentation, Anwendungslogik und Speicherung jeweils in eine eigene Schicht separiert.

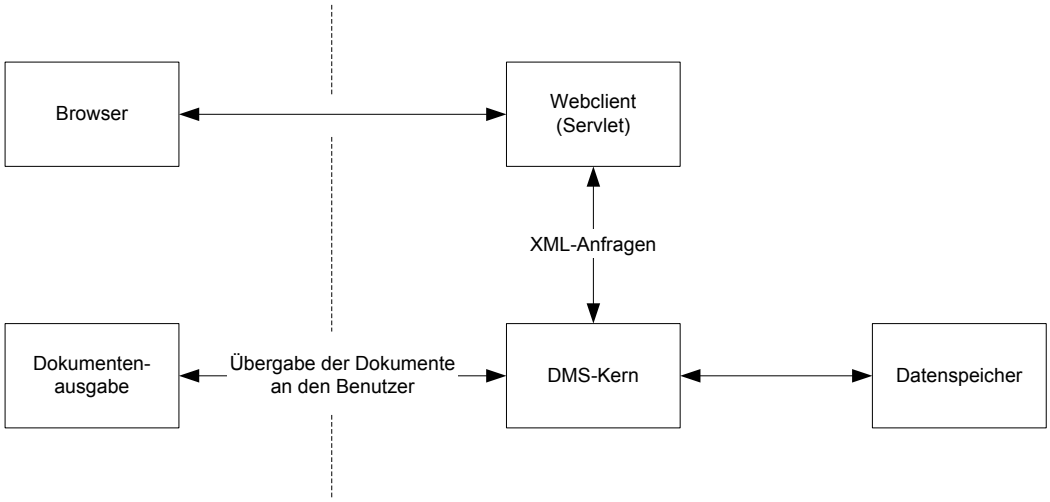

Abbildung 3.1: Grobstruktur des Dokumentenmanagementsystems, Quelle: Eigene Darstellung

Diese Dreiteilung hat den Sinn, dass einzelne Teile ausgetauscht werden können. Das ist notwendig, um zum Beispiel unterschiedliche Datenbanken oder andere Dienste zur Speicherung von Dokumenten, Nutzerberechtigungen und zum Zugriff auf Benutzerbereiche zu integrieren. Weiter sind unterschiedliche Anwenderschnittstellen möglich. So kann der Web-Client durch einen betriebssystemspezifischen Client ersetzt werden, der beispielsweise in den Explorer von Windows integriert wird.

3.1.1 Web-Client

Die oberste Schicht, also die Schnittstelle zum Benutzer, bildet ein Web-Client. Dieser Web-Client wird in dem Webbrowser des Anwenders angezeigt und bietet die Schnittstelle zur Bedienung des Dokumentenmanagements. Diese Vorgehensweise hat mehrere entscheidende Vorteile, die in Abschnitt 3.2.1 genauer diskutiert werden.

3.1.2 Dokumentenmanagementsystem-Kern

Der Kern des DMS konzentriert die Anwendungslogik und übernimmt somit die Organisation der Dokumente. Er verfügt über drei Arten von Schnittstellen.

Die Schnittstelle zur Anwendung wird über eine Netzwerkverbindung realisiert. Das System kann mit Befehlen aus der Anwendung komplett gesteuert werden, das heißt, es können Informationen abgerufen, Aktionen ausgelöst und das System administriert

werden. Dazu wird eine XML-Sprache verwendet (mehr dazu in den Abschnitten 3.2.4 und 3.3.2).

Die Mechanismen zur Abarbeitung der erhaltenen Befehle erlauben die Integration beliebiger Module. Dazu gibt es eine Schnittstelle für abarbeitende Module, die genutzt werden kann, um die Funktionalität des Systems zu erweitern. Dadurch soll eine sinnvolle Gliederung der einzelnen Funktionsgruppen wie beispielsweise Kernfunktionen, Retrieval oder Groupware ermöglicht werden.

Die dritte Art von Schnittstellen sind die Datenspeicherschnittstellen. Den Modulen der Befehlsabarbeitung wird dadurch die Möglichkeit gegeben, modulspezifische Daten zu speichern.

3.1.3 Datenspeicher

Unter dem Begriff Datenspeicher werden mehrere Schnittstellen zur Speicherung von Daten zusammengefasst. Es wird zwischen dokument-, benutzer- und systemspezifischen Daten unterschieden. Dokumentspezifische Daten sind Metadaten, die einem Dokument zugeordnet werden können, wie zum Beispiel Schlagwörter, oder die Binärdaten des Dokuments selbst. Benutzerspezifische Daten werden Benutzern des Systems zugeordnet, beispielsweise die Liste der Arbeitsaufgaben. Zu den systemspezifischen Daten gehören Konfigurationen des DMS und seiner Module.

Die einzelnen Arten von Daten können mit unterschiedlichen Systemen gespeichert werden. Dokumentspezifische oder systemspezifische Daten können als Binärdaten in Datenbanken oder Dateisystemen gespeichert werden (siehe Abschnitt 2.4.5). Benutzerspezifische Daten können in den unterschiedlichsten Systemen gespeichert werden (siehe Abschnitt 2.4.1).

3.1.4 Dokumentenausgabe

Die Dokumentenausgabe subsummiert alle vom DMS angebotenen Möglichkeiten, Dokumente zwischen dem System und dem Benutzer auszutauschen. Das Ansehen und Bearbeiten von Dokumenten bedingt den Transfer von Dateien zwischen dem DMS-Server und dem Arbeitsplatzrechner des Anwenders. Wie in Abschnitt 2.4.4 erörtert, gibt es viele Möglichkeiten zum Transport von Dateien durch Netzwerke.

Um eine maximale Kompatibilität zu erreichen, hat das DMS eine Schnittstelle, die Module für mehrere Protokolle unterstützt. Der Anwender kann im Web-Client wählen, wie er auf Dokumente zugreifen möchte. So wird die Integration in alle Betriebssysteme erreicht und Zukunftssicherheit gewonnen. Eine weitere Diskussion dieses Themas findet im Abschnitt 3.2.3 statt.

3.1.5 Beispiel

Nun soll die Struktur des DMS anhand eines einfachen Beispiels anschaulich dargestellt werden. Die Details zur Benutzerschnittstelle werden ausführlich in Abschnitt 3.6 diskutiert.

Wenn ein Anwender ein Dokument lesen möchte, öffnet er zunächst seinen Webbrowser mit der Startadresse des DMS. Dort recherchiert er nach dem Dokument und bekommt eine Seite mit den entsprechenden Suchergebnissen. Nun kann er *Dokument lesen* auswählen. Der Web-Client fragt nun beim DMS-Kern an, welche Optionen zur Übergabe der Daten an den Benutzer zur Verfügung stehen und welche zusätzlichen Eingaben dazu notwendig sind. Der Anwender wählt als Art der Datenübergabe *WebDAV* aus. Er gibt das Passwort für den WebDAV-Zugang ein, sein Benutzername wird automatisch als Account-Kennung verwendet. Der Web-Client sendet an das Kernsystem die ID des Dokuments und den Übergabemodus samt Parametern im Befehl zum Abruf des Dokument.

Das Kernsystem prüft die Berechtigung des Benutzers auf Lesezugriff und kopiert die Dateien aus dem Dokumentdatenspeicher in ein Verzeichnis des WebDAV-Account des Anwenders. Nachdem der entprechende Zugriff in den Metadaten des Dokuments und des Benutzers vermerkt ist, ist der Vorgang abgeschlossen und der Web-Client erhält eine entsprechende Bestätigung.

Der Web-Client zeigt dem Anwender nun an, dass das Dokument verfügbar ist.

3.2 Auswahl der Technologien

3.2.1 Web-Client

Der wichtigste Vorteil eines Web-Clients, also von Anwendungen, die auf Webservern ausgeführt werden, ist die Plattformunabhängigkeit der Anwendung auf Benutzerseite.

Der Benutzer benötigt zur Benutzung nur einen Webbrowser. Webbrowser sind für nahezu alle Plattformen erhältlich [86] [87] [88]. Die Frage wie die Dokumente auf den Rechner des Benutzers kommen, wird weiter unten geklärt. Dazu eigenen sich Browser beziehungsweise Web-Clients nicht sehr gut, aber eine Notlösung ist trotzdem über die Upload- und Download-Funktionen des HTTP-Standards [89, S. 53–55] möglich.

Der nächste Vorteil ist die Vereinfachung der Durchführung von Software-Updates und Fehlerbehebungen. Da die Anwendungslogik im Web-Client zentral auf einem Serversystem gespeichert ist, muss nur dort – und nicht auf allen Arbeitsplatz-Computern – die neue Version eingespielt werden.

Ebenso von Vorteil ist, dass die Secure-Socket-Layer-(SSL)- [90] bzw. Transport-Layer-Security-(TLS)-Funktionalität [91] der Browsersoftware verwendet werden kann, um die übertragenen Daten zu verschlüsseln. Damit wird der Datenschutz deutlich verbessert. Weiter kann mit TLS/SSL eine Authentifizierung der Clients und des Servers mit entsprechenden Zertifikaten durchgeführt werden, so dass nur berechtigte Anwender auf die Software zugreifen können.

Ein nicht zu unterschätzender Vorteil ist, dass die Arbeitsweise mit Webbrowsern den meisten Anwendern bekannt ist. Es ist eine kürzere Einarbeitungszeit zu erwarten, da die Navigation wie auf Websites erfolgt. Weiter kann der Web-Client in eine bereits vorhandene Intranetlösung wie ein Unternehmensportal leicht integriert oder das Design nach den Wünschen des Kunden angepasst werden.

3.2.2 Java

Als Entwicklungsplattform für das DMS wurde Java [92] gewählt. Java bietet mit der Technik der Virtual Machine [93] eine einfache Möglichkeit, die erstellte Software auf viele Plattformen zu portieren. Dazu müssen lediglich die entsprechenden Laufzeitumgebungen (Runtime Environments) vorhanden sein. Für die Systeme Windows, Linux, Solaris, MacOS und andere UNIX-Derivate sind diese verfügbar [94]. Damit kann auch serverseitig ein breites Spektrum an Plattformen unterstützt werden.

Neben der Plattformunabhängigkeit bietet Java weitere entscheidende Vorteile. Java bietet in der Enterprise Edition [95] eine Vielzahl von offenen Programmierschnittstellen (APIs), die die Anbindung von Softwaresystemen erlauben [96] [97]. Die relevanten APIs werden in den folgenden Unterabschnitten diskutiert. Alle diese Schnittstellen sind frei verfügbar und ohne Entrichtung von Lizenzgebühren nutzbar [98].

Java bietet auch für viele Protokolle Standardimplementierungen an. So werden SSL [90] beziehungsweise TLS [91] und XML (siehe Abschnitt 3.2.4) unterstützt [99, S. 112]. Implementierungen der Applikationsserver für die in der Java 2 Platform Enterprise Edition Specification beschriebenen APIs [99, Kap. 6] sind sowohl von kommerziellen Herstellern als auch im Open Source Bereich verfügbar [100]. So können zusätzliche Lizenzkosten für Applikationsserver und Programmbibliotheken vermieden beziehungsweise gesenkt werden.

3.2.2.1 Servlets

Die zentrale Komponente des DMS, der Dispatcher, wird als Servlet implementiert. Servlets sind Java-Programme, die nach der Java Servlet Specification [101] Server-Funktionalität nach dem Anfrage/Antwort-Schema (Request/Response) bereitstellen. Sie implementieren die Servlet-Schnittstelle und werden in einem Container (Applikationsserver) ausgeführt, der die nicht anwendungsspezifischen Funktionen des Server-Programms übernimmt. Dadurch ist es nicht notwendig, die kommunikations- und betriebssystemspezifische Funktionalität zu programmieren, wodurch Plattformunabhängigkeit und Arbeitsersparnis erzielt werden.

Neben kommerziellen stehen auch nicht-kommerzielle Container zur Verfügung [102] [103], wodurch eine breite Unterstützung der Technologie vorhanden ist.

3.2.2.2 Enterprise JavaBeans

In einem DMS werden ähnlich wie in Datenbankmanagementsystemen Transaktionen benötigt, um die Geschäftsprozesse abbilden zu können. Das heißt, Vorgänge wie *Dokument speichern* sollen nach dem ACID-Prinzip, also atomar, konstistent, isoliert und dauerhauft, durchgeführt werden, damit das System stets korrekt arbeitet. Zur Lösung dieser Problematik wurden zunächst Transaction-Processing-Monitore (TP-Monitore) und später Komponenten-Transaktionsmonitore (CTM), eine spezielle Art Applikationsserver, entwickelt [104, S. 14–17].

Enterprise JavaBeans (EJB) ist ein serverseitiges Standard-Komponentenmodell für Komponenten-Transaktionsmonitore [104, S. 5]. Die Verwendung eines CTM hat den Vorteil, dass sie als Hybrid aus Object-Request-Brokern und TP-Monitoren verteilter objektorientierter Systeme sind, womit es möglich wird, die Dokumente als Objekte zu verwalten, was dem objektorientierten Entwurf sehr entgegenkommt.

Durch die Standardisierung der Enterprise JavaBeans [105] können verschiedene Applikationsserver eingesetzt werden, die je nach benötigter Leistungsfähigkeit ausgewählt werden können. Es existieren sowohl kostenlose CTMs wie JBoss [106], aber auch sehr leistungsfähige kommerzielle CTMs [100].

3.2.2.3 JDBC

Für den Zugriff auf Datenbanken wird das JDBC Data Access API [107] verwendet. Dieses API stellt Klassen zum Arbeiten mit Datenbanksystemen bereit, die unabhängig vom jeweiligen Datenbanksystem sind. Darin sind alle notwendigen Funktionen enthalten, um mit der Datenbank arbeiten zu können, das heißt Funktionen zur Ausführung von SQL-Anfragen [108] und zur Auswertung der Ergebnisse. JDBC-Treiber sind für nahezu alle Datenbanken erhältlich [109]. Es gibt oft direkt vom jeweiligen Datenbankhersteller JDBC-Treiber, aber auch Drittanbieter liefern Treiber, die für mehrere Datenbanksysteme einsetzbar sind. Somit ist es gegebenenfalls möglich, die Treiber ohne Änderungen der Software auszutauschen.

3.2.3 WebDAV

Das DMS wird zum Datenaustausch mit dem Benutzer mehrere verschiedene Optionen anbieten, damit die in Abschnitt 2.4.4 genannten Forderungen bezüglich Kompatibilität und Einfachheit des Transports der Dokumente zum Benutzer erfüllt werden.

Eines der unterstützten Protokolle wird WEBDAV sein. WEBDAV selbst hat eine gute Unterstützung seitens verschiedener Applikationen und Betriebssysteme (Abschnitt 2.4.4). Die WEBDAV-Unterstützung wird so realisiert, dass das DMS als Client in einen WEBDAV-Server Dokumente kopieren und auslesen kann. Bei der Übernahme der Daten können die Metadaten (Properties) der WEBDAV-Dokumente und -Kollektionen übernommen beziehungsweise gesetzt werden.

Es stellt sich die Frage, warum keine reine WEBDAV-Schnittstelle zum System bereitgestellt wird, insbesondere da sehr viele der Grundfunktionen eines DMS im WEBDAV- und DeltaV-Standard enthalten sind.

Für die Unterstützung alter Betriebssystemversionen ist ein reines WEBDAV-System ungeeigent, da es sich nicht besonders gut in das Betriebssystem integrieren lässt, andere Anbindungen etwa über NFS oder FTP dagegen gut funktionieren. Ein weiteres Problem

von WEBDAV ist die Versionierung, die zu Applikationen kompatibel sein muss, die keine Versionierung durchführen. Dazu muss beim Zugriff transparent eine neue Version generiert werden [59]. Dieses Vorgehen würde die Qualität des Dokumentenmanagements zu stark absenken. Weiter müsste das Zusammenführen von Versionen möglich sein, was aber – wie in Abschnitt 2.2.2.2 diskutiert – nicht möglich ist. Schließlich ist auch die Verwaltung der Benutzerrechte noch ungenügend gelöst (ebenfalls in Abschnitt 2.4.4 erläutert).

3.2.4 XML

Die Speicherung der Metadaten der Dokumente und die Kommunikation mit dem DMS werden in einem XML-Format realisiert. XML [110] findet eine breite Unterstützung in vielen Anwendungen, Programmiersprachen und Plattformen [111]. Mit der Definitionssprache XML Schema [112] lassen sich Datenformate definieren und mit geeigneten Parsern wie [113] auf Korrektheit prüfen. Durch XML Namespaces [114] lassen sich Erweiterungen der definierten XML-Sprachen gut implementieren. Vertiefende Informationen zum Themenkomplex XML finden sich in [115] [111].

3.3 Kernsystem

3.3.1 Aufbau

In Abbildung 3.2 ist die Architektur des DMS detailliert dargestellt. Das Kernsystem besteht aus mehreren Teilen, nämlich dem Dispatcher, dem Core-DM-Modul, der Authentifizierung, dem User-Management-Modul, der User- und Document-Storage-Schnittstelle, der Input- und Output-Modul-Schnittstelle und dem Task-Scheduling-Service. Alle weiteren Module gehören nicht zum Kernsystem und sind optional.

Die Eingabedaten, also die Befehle an das DMS, werden in einer XML-Sprache formuliert und die Ausgabedaten, das heißt die Antworten auf diese Befehle, werden ebenfalls mittels einer XML-Sprache formuliert. Zusätzlich können Eingaben und Ausgaben Dokumentteile enthalten, die über die Input- und Output-Modul-Schnittstelle ins System gelangen.

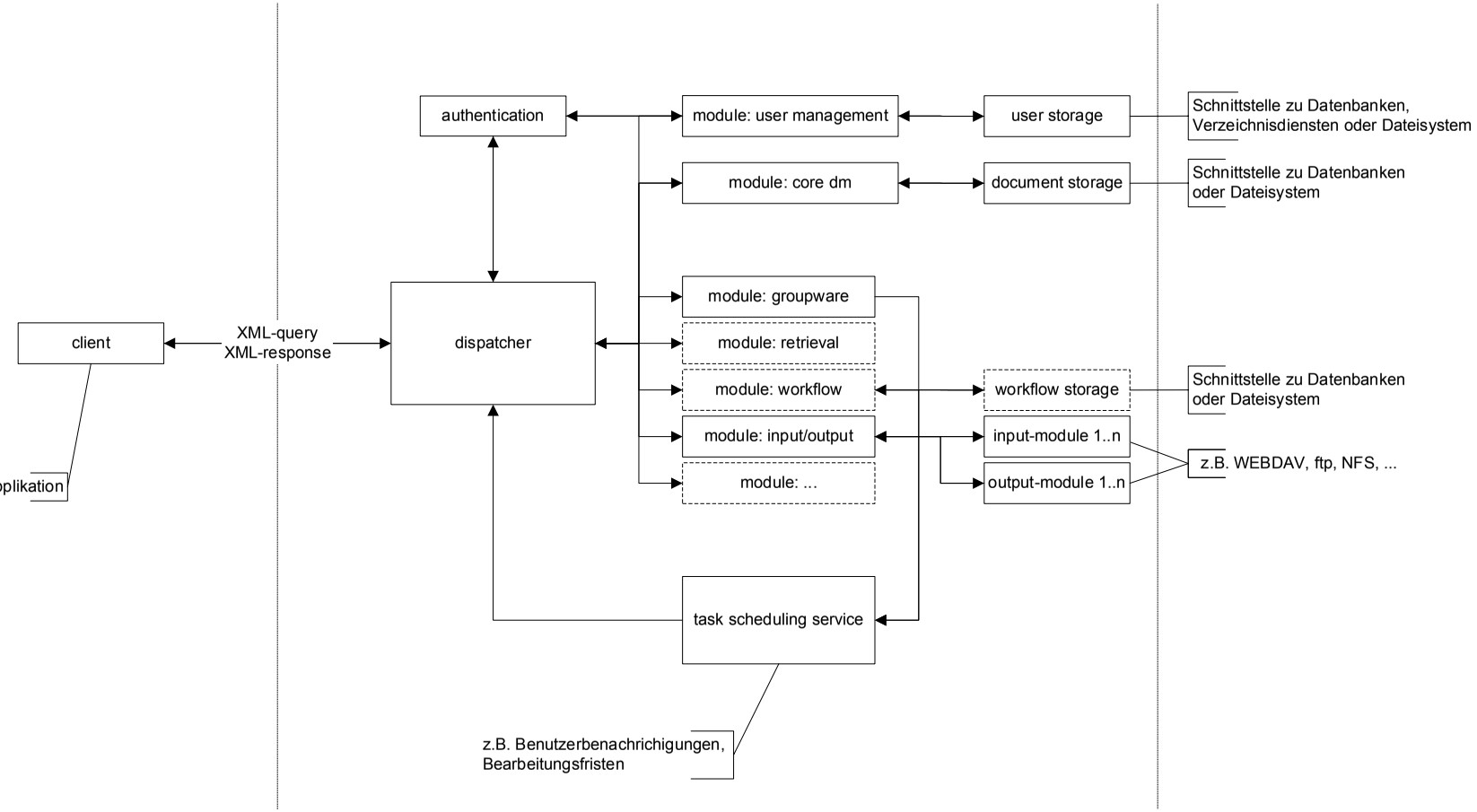

Abbildung 3.2: Architektur des Dokumentenmanagementsystems, Quelle: Eigene Darstellung

3.3.2 Dispatcher

Der Dispatcher ist die Komponente, mit der die Client-Anwendung kommuniziert. Er ist für die Annahme, Weiterleitung und die Antwort auf Anfragen zuständig.

Die Realisierung erfolgt als Servlet, so dass bei der Implementierung des Dispatchers keine Low-Level-Funktionen zur Kommunikation über TCP/IP-Netzwerke programmiert werden müssen, da diese Aufgaben vom Servlet-Container übernommen werden.

3.3.2.1 Annahme von und Antwort auf Anfragen

Dem Dispatcher wird eine vollständige XML-Anfrage übergeben, die er mittels eines XML-Parsers in eine DOM-Repräsentation [116] überführt. Aus dieser DOM-Repräsentation extrahiert er zunächst die Benutzerdaten zur Authentifizierung und Autorisierung und übergibt sie an das Benutzermodul (siehe Abschnitt 3.4.2).

Nach erfolgreicher Prüfung muss der Dispatcher die enthaltenen Anfragen, die in Teilbäumen des DOM-Dokuments repräsentiert sind, an die entsprechenden Module weiterverteilen (siehe Abschnitt 3.3.2.2). Die Antworten der Module werden vom Dispatcher gesammelt und in den DOM-Baum der Antwort integriert.

Sobald alle Anfragen verarbeitet sind, sendet der Dispatcher den serialisierten Antwort-DOM-Baum an den Client zurück und schließt die Verbindung. Im Falle von Fehlern ist es die Aufgabe des Dispatchers, eine Fehlermeldung zu generieren. Für die Ebene der Applikationsfehler hingegen sind die einzelnen Module zuständig.

3.3.2.2 Verteilung der Anfragen

Zur Verteilung der gestellten Anfragen muss der Dispatcher die zuständigen Module ermitteln. Dazu führt er eine Liste, worin sich die Module bei Start des Systems registrieren können, die einem bestimmten XML-Namensraum (Namespace) Module zuordnet. Auf einen Namensraum können sich mehrere Module registrieren. Für jedes auftretende Element werden die auf seinen Namensraum registrierten Module in umgekehrter Reihenfolge der Registrierung aufgerufen.

3.3.2.3 Kommunikation mit den Modulen

Die Eingabedaten werden den Modulen in Form von DOM-Bäumen übergeben, die als Wurzel ein DOM-Element haben. Die Module können das Element, das die Anfrage repräsentiert, und seinen Unterbaum gegebenenfalls modifizieren. Darüber hinaus ist es den Modulen möglich, neue Anfragen an den Dispatcher einzufügen, so dass ein Erweiterungsmodul nach der Bearbeitung einer Anfrage mehrere Befehle anderer Module aufrufen kann. Der Dispatcher erhält von den Modulen die Antworten auf die Anfragen als DOM-Elemente zurück, so dass er sie in das Antwortdokument einbetten kann.

Weiter stellt der Dispatcher den Modulen die anderen Dienste des Kernsystems zur Verfügung, so etwa den Zugriff auf die Autentifizierung, die Authorisierung von Zugriffen und den Task-Scheduling-Service.

3.3.3 Task-Scheduling-Service

Der Task-Scheduling-Service ist für Anfragen zuständig, die zu bestimmten Zeitpunkten ausgeführt werden müssen. Er ist notwendig, da das DMS nach dem Anfrage-Antwort-Prinzip arbeitet und die Abarbeitung so schnell wie möglich durchgeführt wird, aber einige Aufgaben zu späteren Zeitpunkten ausgeführt werden sollen. Durch den Task-Scheduling-Service wird den Modulen eine Schnittstelle zur Verfügung gestellt, die Abarbeitung von beliebigen Anfragen zu gewünschten Zeitpunkten ermöglicht.

Im Task-Scheduling-Service wird eine chronologische Liste der auszuführenden Anfragen gehalten. Zum Ausführungszeitpunkt nimmt der Task-Scheduling-Service eine Verbindung zum Dispatcher auf und arbeitet die Anfragen ab. Zum Zugriff auf die geplanten Anfragen generiert der Server eine eindeutige Identifikationsnummer, die dem aufrufenden Modul zum Zugriff auf die geplante Aufgabe zur Verfügung gestellt wird. Dadurch kann ein Modul zu einem späteren Zeitpunkt eine Aufgabe aus der Liste des Servers entfernen, das Ergebnis abrufen oder die Aufgabe modifizieren.

3.3.4 Module

Zur Erhöhung der Flexibilität des DMS werden die Funktionen bestimmten Gruppen zugeordnet, die dann von Modulen bearbeitet werden. Ein Modul registriert sich beim Dispatcher für eine oder mehrere Funktionsgruppen, die durch Namensräume innerhalb der XML-Anfragen festgelegt werden.

Durch die Modulschnittstelle wird das System für Erweiterungen geöffnet, wie dies in Kapitel 2 mehrfach gefordert wurde. Erweiterungsmodule können auf die Funktionalität des Kernsystems zugreifen und erhalten dadurch vollen Zugang zur Grundfunktionalität des DMS.

3.3.4.1 Extraktion von Anfragen

Die Anfragen werden wie oben (Abschnitt 3.3.2.2) beschrieben durch den Dispatcher als Teil eines DOM-Baums an das Modul geliefert. Die Module können einen eigenen Befehlssatz verwenden, das heißt die XML-Struktur frei definieren – unabhängig von allen anderen Modulen. Ebenso können sie bereits vordefinierte Strukturen anderer Module erweitern und diese DOM-Teilbäume gegebenenfalls für den Aufruf der anderen Module modifizieren.

3.3.4.2 Speicherung von Einstellungen

Jedem Modul wird vom Dispatcher eine Schnittstelle zum Zugriff auf die Dokumente, die benutzerbezogenen und globalen Daten zur Verfügung gestellt. Durch diese Schnittstelle kann ein Modul eigene Daten zum Dokument oder auch nur zu Teilen des Dokuments speichern, etwa eine Schlagwortliste. Genauso kann das Modul benutzerbezogen Konfigurationen speichern. Für globale Konfigurationen, die für alle Benutzer und Dokumente relevant sind, kann ein Modul beliebig viele Daten speichern.

3.3.4.3 Reaktionsmöglichkeiten

Jedes Modul hat bei der Abarbeitung einer Anfrage mehrere Möglichkeiten zur Reaktion:

1. Bei korrekter Abarbeitung der Anfrage das Generieren einer entsprechenden Antwort als DOM-Teilbaum,
2. bei fehlerhafter Abarbeitung das Erzeugen einer Antwort als DOM-Teilbaum, der auf Anwendungsebene als Fehlermeldung interpretiert und an den Benutzer weitergegeben wird,
3. bei fehlerhafter Abarbeitung das Stoppen einer weiteren Verarbeitung des aktuellen Befehls durch andere Module,

4. bei fehlerhafter Abarbeitung die Erzeugung einer globalen Fehlermeldung und den Abbruch der Abarbeitung sämtlicher Befehle.

Die letzten beiden Möglichkeiten zur Reaktion sollen so sparsam wie möglich verwendet werden, insbesondere das Abbrechen weiterer Verarbeitung.

3.4 Kernmodule

Dieser Abschnitt beschreibt die in den einzelnen Modulen des Kernsystems realisierte Funktionalität. Das Kernsysteme muss alle Funktionen eines allgemeinen DMS abdecken können (siehe Abschnitt 2.2), abgesehen von den Anforderungen bezüglich des Retrieval (Abschnitt 2.2.3) – dabei muss nur das Auffinden nach einfachen Kriterien (Name des Dokuments, Autor, usw.) möglich sein.

3.4.1 Kernfunktionen des Dokumentenmanagements

Die Dokumente werden mit einer Dokumenten-Identifikationsnummer (Dokumenten-ID) versehen. Über diese Nummer kann auf die Dokumente zugegriffen werden. Zusätzlich werden zu einem Dokument ein Name, eine Beschreibung, der Benutzer, der das Dokument erstellt hat und das Datum der Erstellung gespeichert. Zusätzliche Metadaten sind dem Web-Client beziehungsweise den Anwendungen überlassen und können als beliebige Binärdaten gespeichert werden. Weiter wird jedem Modul erlaubt, zu einem Dokument beliebig viele zusätzliche Datenblöcke zu speichern.

Im Dokumentenmanagementsystem werden zu einem Dokument die Dateien als Menge von Binärdatenblöcken gespeichert, so dass eine Version einer Datei im Dokument einem Binärdatenblock im DMS entspricht. Zusätzlich werden sämtliche Daten zur Strukturierung, das heißt die Verzeichnisstruktur, die Namen der Dateien, die Änderungsdaten der Dateien und die Versionsinformationen, in einem XML-Datenformat gespeichert. In Abbildung 3.3 wird dieses System graphisch dargestellt.

Das Dokument besteht aus verschiedenen Versionen. Von Version zu Version können sich die Daten ändern. Folgende Punkte müssen bei der Speicherung des Dokuments beachtet werden:

- Dateien oder Verzeichnisse sind hinzugekommen
- Dateien oder Verzeichnisse sind entfernt worden

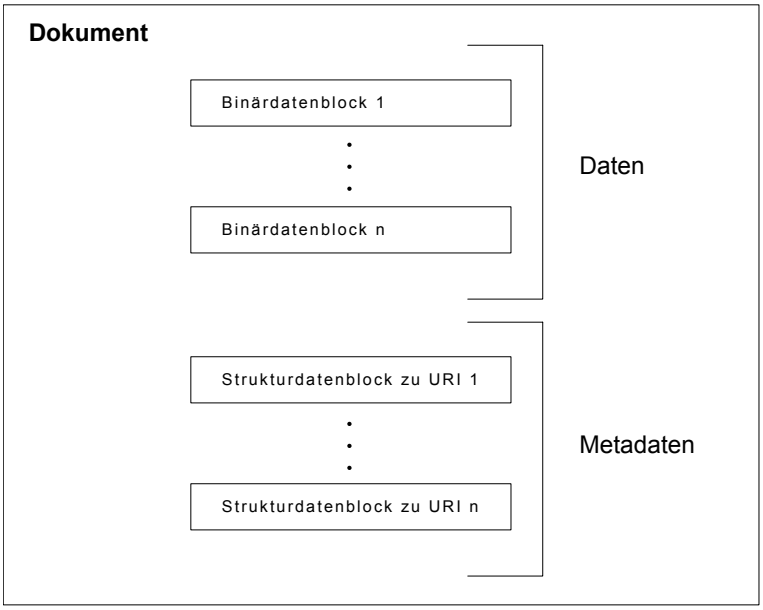

Abbildung 3.3: Interne Struktur eines Dokuments, Quelle: Eigene Darstellung

- Dateien oder Verzeichnisse wurden verändert
- Dateien oder Verzeichnisse sind unverändert

Bei der Speicherung einer Version muss also ein Vergleich zur Vorgängerversion durchgeführt und entsprechend auf die obigen Punkte reagiert werden. Sind Dateien aus der Vorgängerversion unverändert geblieben, müssen sie nicht neu abgespeichert werden. Anderfalls muss ein neuer Binärdatenblock angelegt werden, ebenso bei hinzugekommenen Dateien. Beim Entfernen von Dateien müssen nur die Strukturinformationen angepasst werden.

Zu den Dateien müssen Strukturinformationen gespeichert werden. In Abbildung 3.4 ist zu erkennen, wie die Struktur eines Dokumente abgespeichert wird. Zu jeder Version des Dokuments wird die komplette Verzeichnisstruktur inklusive aller Unterverzeichnisse abgespeichert. Den Dateien werden dann Datenblöcke zugeordnet. Versionen von Dateien werden immer nur einmal abgespeichert, das heißt Dateien, die sich zwischen Versionen nicht verändern, werden nicht kopiert (siehe Beispiel in Abbildung 3.4).

Zu den Dateien und Verzeichnissen werden zusätzlich Metadaten erfasst. Dabei sind die vorgegebenen Metadaten bewusst minimal gehalten. Sie umfassen den Namen der Datei oder des Verzeichnisses, eine Beschreibung und den Bearbeiter, der die Datei oder das Verzeichnis gespeichert hat. Für andere Metadaten sind die Anwendungen selbst

beziehungsweise der Web-Client zuständig. Es können zusätzlich Metadaten in Form von Binärdaten gespeichert werden.

Die Dokumente werden im System in Form von Entity Beans verwaltet, damit alle Modifikationen an den Dokumenten in Form von Transaktionen durchgeführt werden können. Generell haben alle Module direkten Zugriff auf die Beans. Sie sollten aber soweit wie möglich die Schnittstellen des Dispatchers nutzen, um auf die Dokumente zuzugreifen.

Eine Liste der Funktionen des Moduls findet sich im Anhang A.1.

3.4.2 Benutzerverwaltung und Authentifizierung

3.4.2.1 Aufbau der Benutzerverwaltung

Die Benutzerverwaltung erfüllt die Aufgaben der Authentifizierung und der Authorisierung. Die Authentifizierung, also die eindeutige und sichere Identifikation des Benutzers, wird durch den Dispatcher nach dem Einlesen und Parsen des Requests durchgeführt. Die Client-Anwendung übermittelt dazu in den XML-Daten eine Authentifizierungsanfrage. Für die Authentifizierung wird der zugehörige Teilbaum des DOM-Dokuments an das Benutzerverwaltungsmodul übergeben. Es gibt eine Reihe von unterschiedlichen Verfahren zur Authentifizierung [117], etwa durch simple passwortgeschützte Benutzerkonten wie bei Betriebssystemen oder durch kompliziertere kryptographische Verfahren wie RSA [118]. Das DMS hat eine einfache Authentifizierungsmöglichkeit über Benutzer-Accounts und zugehörige Passwörter, kann aber durch Anbindung an andere Systeme um andere Protokolle erweitert werden.

In der Benutzerverwaltung des DMS gibt es Benutzer und Gruppen. Für jeden Benutzer beziehungsweise für jede Gruppe können die Module Eigenschaften definieren, womit sie den Zugriff auf ihre Funktionen erlauben oder verbieten können. Über Gruppenzugehörigkeiten können auch Rollen [119] simuliert werden, die ein Benutzer besitzen kann. Für Optionen, die nur im Kontext eines bestimmten Dokuments oder eines Teils eines Dokuments sinnvoll sind, kann eine Zugriffsrechteliste speziell für ein Dokument abgespeichert werden. Dazu greift das Benutzerverwaltungsmodul auf die Dokumente direkt zu.

Die Benutzer und Gruppen werden mit Entity Beans modelliert und Zugriffe auf die Rechte werden als Transaktion durchgeführt.

Eine Liste der Funktionen des Moduls findet sich im Anhang A.2.

3.4.2.2 Anbindung an externe Systeme

Das Benutzerverwaltungsmodul kann neben der internen Verwaltung der Rechte über eine Schnittstelle auf externe Systeme wie beispielsweise den Verzeichnisdienst LDAP zugreifen. Dabei werden die Daten soweit möglich auf dem externen System gespeichert. Ebenso ist ein Mischbetrieb möglich, so dass nur lesend auf die externen Systeme zugegriffen wird, um die Benutzerlisten des Betriebssystems oder des Verzeichnisdienstes auf die intern verwalteten Benutzer abzubilden und die Authentifizierung extern durchzuführen.

3.4.3 Ein- und Ausgabe

Die Teile des DMS, die der Ein- und Ausgabe der Dokumente vom beziehungsweise zum Benutzer dienen, sind modular aufgebaut. Damit ist es möglich, zwischen verschiedenen Arten des Transfers zu wählen. Beispielsweise kann der Nutzer beim Laden eines Dokuments zwischen seinem Benutzerverzeichnis auf dem Datei-Server, seinem WEBDAV-Ordner und einem FTP-Konto wählen. Die dazu notwendigen Daten wie zugehörige Benutzernamen und Passwörter werden dem Ein- oder Ausgabemodul als Parameter übergeben.

Die vorhandenen Module werden beim Start des DMS geladen und können über eine Schnittstelle des Dispatchers angesprochen werden.

Grundsätzlich sind Ein- und Ausgabemodul getrennt, da manche Systeme nur Ein- oder nur Ausgabe ermöglichen. So würde eine Abrufmöglichkeit über das HTTP-Protokoll als Ausgabemodul implementiert werden. Ein Modul implementiert eine Schnittstelle, die die Struktur eines Dokuments möglichst genau auf dem Zielsystem abbilden kann.

Einige Systeme bieten die Möglichkeit, Metadaten zu Dateien zu speichern, so etwa WEBDAV. Daher unterstützen die Ein- und Ausgabemodule über die Schnittstelle gegebenenfalls auch den Transfer von Metadaten.

3.4.4 Datenspeicherung

Die Speicherung der Daten des DMS, genauer also der Dokumente, Benutzer (sofern nicht in Fremdsystemen) und Konfigurationsdaten sowie der Daten der spezialisierten Module, wird über eine allgemeine Schnittstelle zur Datenspeicherung durchgeführt.

Grundsätzlich sind die Module für ihre eigenen Daten verantwortlich. Die Kernmodule verwenden zur Speicherung Binärdaten für die Dateien der Dokumente und XML-Datenformate zur Speicherung der eigenen Verwaltungsdaten wie etwa für die Dokumentstruktur. Daher müssen über die Schnittstelle beliebige Datenblöcke gespeichert und geladen werden können.

Eine Abbildung auf ein Datenbankschema würde die in Abschnitt 2.4.5 geforderte Möglichkeit zur (optionalen) Verwendung von Dateisystemen als Speicherort unmöglich machen. Daher werden einfach Binärdatenblöcke mit numerischer Identifikation über die Schnittstelle verwendet. Die Schnittstelle kann für jedes Modul eigene Datenblöcke speichern und laden.

Die Schnittstelle ist so aufgebaut, dass beliebige Systeme zur Speicherung verwendet werden können. Die beiden gängigsten sind Datenbanksysteme, die über die JDBC-Schnittstelle angebunden werden, und Dateisysteme. Bei der Speicherung in Dateisystemen werden die Blöcke direkt auf Dateien in einer sinnvoll strukturierten Verzeichnisstruktur und in Datenbanksystemen auf Binary Large Objects (BLOBs) umgesetzt.

Die Definition der Funktionen findet sich im Anhang A.3.

3.5 Funktionsmodule

In den anderen Modulen werden alle höheren Funkionalitäten implementiert, die über die grundlegenden Funktionen der Kernmodule hinausgehen. Diese Module verwenden die Befehle der Grundmodule. Dazu können sie – wie bereits in Abschnitt 3.3.4.3 beschrieben – die XML-Bäume der Befehle modifizieren. Darüberhinaus können sie auch eigene Daten speichern und auf den Dispatcher und den Task-Scheduling-Service zugreifen.

Im Rahmen dieser Diplomarbeit ist das Groupware-Modul von besonderem Interesse. Aus diesem Grund wird nur dieses Modul ausführlich beschrieben, andere Module wie etwa zum Retrieval oder zur Workflow-Verwaltung werden nur kurz erwähnt, sind im eigentlichen System jedoch vorhanden.

3.5.1 Groupware-Modul

3.5.1.1 Übersicht zu den Funktionalitäten des Moduls

Die Funktionalität des Groupware-Moduls kann in vier Kategorien eingeteilt werden.

Awareness: Das Modul bietet umfassende Möglichkeiten zur Überwachung von Dokumenten und der Tätigkeiten anderer Benutzer an. Bei Dokumenten können beliebige Aktionen mit dem Dokument überwacht werden, insbesondere Bearbeitung durch andere Benutzer.

Adhoc-Workflow: Die Funktionen zur Realisierung von Adhoc-Workflows geben die Möglichkeit, bei bestimmten Ereignissen wie dem Zurückspeichern einer neuen Version des Dokuments beliebige Befehle auszuführen. Damit können vom Benutzer selbst Workflows modelliert und ausgelöst werden.

Nachrichten: Das Groupware-Modul ermöglicht es den Benutzern, Nachrichten zu senden, die die Bearbeitung von Dokumenten betreffen. Es gibt eine Reihe von typischen Nachrichten als Vorlage (Bearbeitungswunsch, Referenz, Informationsanfragen), es können aber auch eigene Nachrichten versandt werden. Nachrichten können nur in Bezug auf Dokumente erstellt werden.

Diskussion: Zu jedem Dokument stellt das Modul ein Diskussionsforum bereit, in dem Benutzer nach Themen sortiert über das Dokument diskutieren oder Fragen stellen können. Diese Funktionalität soll die Absprache unter räumlich getrennten Bearbeiter, insbesondere bei unterschiedlichen Arbeitszeiten (etwa durch verschiedene Zeitzonen) ermöglichen.

Die Funktionen sind im Anhang A.4 spezifiziert.

3.5.1.2 Benutzergesteuerte Aktivitäten

Bei benutzergesteuerten Aktivitäten ist im Gegensatz zu zeitgesteuerten Aktivitäten immer eine durch einen Benutzer aufgerufene Aktivität vorhanden, das heißt, es wird eine Anforderung vom Web-Client an das DMS respektive den Dispatcher gestellt.

Grundsätzlich lösen alle Funktionen der Kernmodule benutzergesteuerte Aktivitäten aus, müssen also vom Groupware-Modul ebenfalls beachtet werden. Aus diesen Grund wird auf die gesonderte Auflistung der Funktionen verzichtet.

3.5.1.3 Zeitgesteuerte Aktivitäten

Als zeitgesteuerte Aktivitäten werden im Groupware-Modul solche Funktionen verstanden, die durch das Erreichen eines Zeitpunktes bestimmte Reaktionen erfordern. Dazu zählt im Wesentlichen das Ablaufen von Fristen. Um diese Aktivitäten zu realisieren,

muss der Task-Scheduling-Service verwendet werden, der – wie in Abschnitt 3.3.3 beschrieben – zu definierten Zeitpunkten Anfragen zur Abarbeitung bringen kann.

3.5.2 Andere Module

Andere Module, die zur Vervollständigung des DMS benötigt werden, sind einmal das sehr wichtige Modul zum Retrieval über dem Dokumentenbestand, das heißt zur Suche im DMS nach den in Abschnitt 2.2.3 genannten Methoden. Für eine einsatzfähige Version des DMS sind Beschlagwortung, Kategorisierung in einer Verzeichnisstruktur und die Volltextsuche notwendig. Entwurf und Implementierung dieser Retrieval-Funktionalität übersteigt aber den Umfang dieser Diplomarbeit und soll nicht weiter betrachtet werden.

Ein weiteres Funktionsmodul ist das Workflowmodul nach den Anforderungen in Abschnitt 2.3.2.2. Für Entwurf und Implementierung gilt dasselbe wie für das Retrieval-Modul.

3.6 Benutzersicht

In diesem Abschnitt soll nicht der Web-Client selbst, sondern nur dessen dem Benutzer angebotene Funktionalität beschrieben werden.

Dem Benutzer wird eine Sicht auf seinen Arbeitsplatz innerhalb des DMS gegeben. Folgende Objekte werden dem Benutzer angezeigt oder zugänglich gemacht:

- Dokumente, die er zum Lesen geöffnet hat, inklusive Optionen,
- Dokumente, die er zum Bearbeiten geöffnet hat, inklusive Optionen,
- Ereignisse, die ihn oder Dokumente, mit denen er arbeitet, betreffen,
- Kontexte zu seinen Gruppenangehörigkeiten, das heißt Benutzer, die mit ihm in einer Gruppe arbeiten, und gemeinsame Dokumentlisten zu den Gruppen,
- Suchfunktionen für die Recherche im Dokumentenbestand,
- Konfigurationsmöglichkeit seiner Sicht auf das System.

Diese Liste erhebt keinen Anspruch auf Vollständigkeit. Sie sollte im Gegenteil ganz speziell nach den Wünschen des jeweiligen Kunden angepasst und erweitert werden.

Dem Benutzer bleibt somit die gesamte interne Struktur verborgen. Seine Sicht ist in Abbildung 3.5 graphisch dargestellt. Er hat nur den Web-Client als zentralen Anlaufpunkt und sein Konto zum Zugriff auf die Dokumente.

Zu einem Dokument oder Teilen eines Dokuments bietet das DMS aus Sicht des Benutzers – soweit relevant – folgende Bearbeitungsmöglichkeiten an:

- Informationen zum Dokument ausgeben (Struktur, Metadaten)
- Status und Geschichte anzeigen (Ereignisprotokoll des Dokuments und aktueller Status hinsichtlich Bearbeitung),
- zum Lesen öffnen, also ohne Möglichkeit der Speicherung von Änderungen, optional Angabe von Version und Teilen des Dokuments,
- zum Bearbeiten öffnen, optional Angabe von Version, Teilen des Dokuments und Beschreibungen der Bearbeitung als Information für andere Benutzer, Bearbeitungsfristen und automatisches Speichern bei Ablauf der Frist sowie Benachrichtigung anderer Benutzer,
- Abbruch der Bearbeitung, optional Angabe einer Begründung und Benachrichtigung anderer Benutzer,
- Dokument speichern, optional Beschreibung der Änderungen und Benachrichtung an vorhergehende Bearbeiter oder andere Benutzer,
- Änderung der Benutzerrechte des Dokuments,
- Ändern der gespeicherten Metadaten,
- Senden an einen Benutzer (zur Bearbeitung), optional Angabe der Aufgabenbeschreibung und Bearbeitungsfristen sowie Reaktion bei Fertigstellung der Aufgabe oder Ablauf der Fristen,
- Kommentieren/Diskutieren im zugehörigen Diskussionsforum, optional Benachrichtigung anderer Benutzer oder Bearbeiter,
- Überwachen, Angabe von relevanten Ereignissen und optionale Angabe von Überwachungszeitraum oder Einschränkung der überwachten Benutzer,
- Anfordern, falls bereits in Bearbeitung, optional Benachrichtigung von Benutzern bei Speicherung und automatisches Laden zum Lesen/Schreiben sowie eine Benachrichtigung des Bearbeiters über die Anforderung,

- Abschließen eines Dokuments, optional Angabe von Aufbewahrungsfristen, automatisches Löschen zu bestimmten Zeitpunkten und Löschen der Geschichte und früherer Fassungen des Dokuments,
- Löschen des Dokuments,
- Hinzufügen zur persönlichen Liste oder zu einer Guppenliste,
- Erstellen eines neuen Dokuments als Kopie.

Zu einem Benutzer bietet das DMS im Falle gegebener Berechtigung des ausführenden Benutzers – soweit sinnvoll – folgende Funktionen an:

- Anzeige von Informationen, optional Anzeige seiner Aufgaben- und Dokumentlisten,
- Ändern der nicht dokumentbezogenen Rechte des Benutzers,
- Hinzufügen einer Gruppenzugehörigkeit,
- Senden einer Nachricht an den Benutzer.

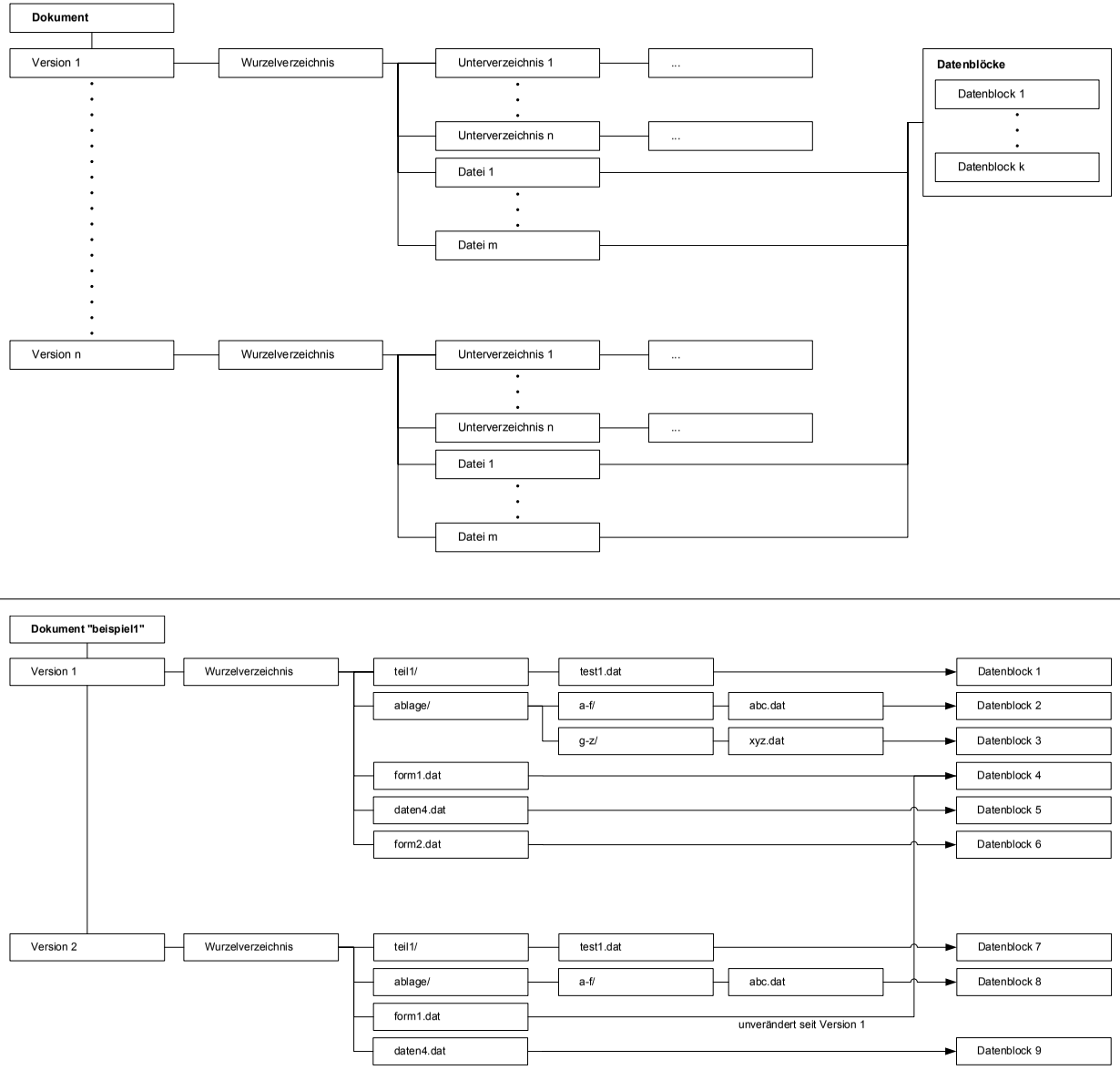

Abbildung 3.4: Struktur eines Dokuments (oben) und Beispiel (unten), Quelle: Eigene Darstellung

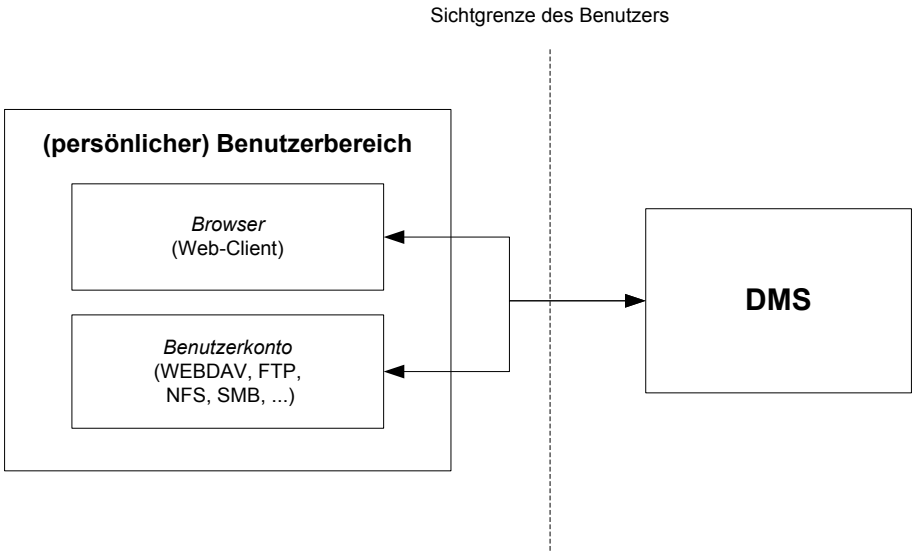

Abbildung 3.5: Sicht des Benutzers auf das DMS, Quelle: Eigene Darstellung

4 Schnittstellen

In diesem Kapitel sollen die Schnittstellen des Systems erläutert werden. Zunächst soll es um die Schnittstellen zum Zugriff von außen gehen (Abschnitt 4.1), die zum Beispiel der Web-Client verwendet. Danach sollen auch die internen Schnittstellen zur Erweiterung des Systems genauer betrachtet werden (Abschnitt 4.2).

4.1 Schnittstellen nach außen

4.1.1 XML-basierter Zugriff der Clients

Der Zugriff von außen wird mittels einer XML-Sprache durchgeführt. Dazu empfängt der Dispatcher den XML-Befehlsblock, das heißt eine XML-Datei. Diese Antwort wird abgearbeitet und mit den Ergebnissen in Form einer zweiten XML-Datei beantwortet. Damit folgt das DMS dem Anfrage/Antwort-Schema. Im folgenden werden die englischsprachigen Begriffe Request für Anforderung und Response für Antwort verwendet.

Die XML-Schemata für die Datenformate sind im Anhang B.1 angegeben.

4.1.1.1 Anfragen

In Abbildung 4.1 findet sich ein Beispiel eines solchen Requests, der durch das Schema `request.xsd` definiert wird.

Das Root-Element eines Requests ist immer das Element `request` (Zeile 1 bis 5), welches zum Namensraum *http://www.ewerk.com/dms/request* gehört. Innerhalb dieses Elements muss als nächstes die Authentifizierung stattfinden, damit der Zugriff auf das DMS authorisiert werden kann. Dazu ist das Element `auth` (Zeile 6) vorgesehen. Es enthält Angaben zur Methode der Authentifizierung (Zeile 7) und die zugehörigen Parameter (Zeilen 8 bis 17). Wird keine Authentifizierung durchgeführt, so ist die Anfrage ungültig und wird mit einer entsprechenden Fehlermeldung beantwortet.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <requests
3      xmlns="http://www.ewerk.com/dms/request"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.ewerk.com/dms/request request.xsd">
6      <auth xmlns="http://www.ewerk.com/dms/auth/request">
7          <method>simple</method>
8          <parameters>
9              <parameter>
10                 <name>user</name>
11                 <value>johndoe</value>
12             </parameter>
13             <parameter>
14                 <name>password</name>
15                 <value>123456</value>
16             </parameter>
17         </parameters>
18     </auth>
19     <request>
20         <list xmlns="http://www.ewerk.com/dms/core/request">
21             <start>0</start>
22             <max>10</start>
23         </list>
24     </request>
25     <request>
26         <message_list xmlns="http://www.ewerk.com/dms/groupware/request">
27             <id>121</id>
28             <only_new>true</only_new>
29         </message_list>
30     </request>
31 </requests>
```

Abbildung 4.1: Beispiel für eine Anfrage an das DMS

Nach der Authentifizierung können dann beliebig viele Anfragen ausgeführt werden. Dazu dient das Element `request` (Zeile 19, 25). Innerhalb dieses Elements wird dann der eigentliche Befehl angegeben, etwa `list` zum Auflisten der vorhandenen Dokumente (Zeile 20) oder `message_list` zum Auflisten der neuen Nachrichten an den Benutzer (Zeile 26). Die Befehle werden durch die Verwendung des entsprechenden Namensraums den einzelnen Module zugeordnet.

Der Dispatcher arbeitet in der verwendeten Reihenfolge die enthaltenen Anforderungen ab, wobei die Gesamtheit der Befehle nicht als Transaktion verarbeitet wird. Nur die einzelnen `request`-Elemente werden als Transaktion durchgeführt.

4.1.1.2 Antworten

Abbildung 4.2 zeigt das Beispiel einer Response, die durch das Schema `response.xsd` definiert wird. Sie bezieht sich auf die im vorhergehenden Abschnitt besprochene Anfrage.

Wurzelelement der Response ist `responses` (Zeile 2). Danach folgt zunächst die Bestätigung `success` (Zeile 6) oder die Ablehnung `failure` der Authentifizierung. Bei erfolgreicher Authentifizierung folgen die Antworten auf die einzelnen Anfragen (`response`).

In diesem Beispiel ist zunächst die erste Anfrage nicht bearbeitet worden, da der Benutzer die Berechtigung für diese Aktion nicht hatte (Zeile 10). Zu Fehlermeldungen werden noch weitere Daten übermittelt: Die Nummer des Fehlers (`code`), die Art des Fehlers (`type`), eine Beschreibung in textueller Form (`description`), das auslösende Modul (`modul`) und der Name der auslösenden Funktion (`function`). Danach wurde noch die Anforderung der neuen Nachrichten für den Benutzer durchgeführt (Zeilen 19 bis 28).

4.2 Schnittstellen nach innen

In diesem Abschnitt werden die Schnittstellen beschrieben, die zur Erweiterung des Systems geschaffen wurden beziehungsweise die Klassen und Schnittstellen, die die Erweiterungsmodule zum Zugriff auf die gespeicherten Daten verwenden können.

Generell gilt für alle Erweiterungen, dass die Module Thread-sicher sein müssen, das heißt, dass sie fehlerfrei unter Mehrfachzugriffen arbeiten sollen. Diese ist notwendig, damit eine effiziente parallele Verwendung der Module möglich wird.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <responses
3      xmlns="http://www.ewerk.com/dms/response"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.ewerk.com/dms/response response.xsd">
6      <auth xmlns="http://www.ewerk.com/dms/auth/request">
7          <success>121</success>
8      </auth>
9      <response>
10         <error xmlns="http://www.ewerk.com/dms/core">
11             <code>1</code>
12             <type>user</type>
13             <description>Der Benutzer hat keinen Zugriff.</description>
14             <module>core</module>
15             <function>list</function>
16         </error>
17     </response>
18     <response>
19         <message_list
20             xmlns="http://www.ewerk.com/dms/groupware/response"
21             xmlns:gdt="http://www.ewerk.com/dms/groupware/datatypes">
22             <messages> <message>
23                 <gdt:sender>122</gdt:sender>
24                 <gdt:date>2001-12-17T09:30:47-05:00</gdt:date>
25                 <gdt:status>unread</gdt:status>
26                 <gdt:subject>Hallo.</gdt:subject>
27             </message></messages>
28         </message_list>
29     </response>
30 </responses>
```

Abbildung 4.2: Beispiel für eine Antwort des DMS

4.2.1 Zugriff auf den Datenspeicher

Der Zugriff auf den Datenbestand ist über Service-Klassen geregelt, auf die alle Module zugreifen dürfen. Instanzen dieser Klassen sind vom Dispatcher erhältlich (siehe Quelltext im Anhang C.7).

Der Zugriff auf den Datenspeicher wird über die Klassen `DocumentStorage` und `ConfigurationStorage` geregelt. Die UML-Klassendiagramme zeigen die Abbildungen 4.3 und 4.4, Quelltexte finden sich im Anhang unter C.1 und C.2. Erstere ermöglicht den Zugriff auf Dokumente, die zweite ist für das Speichern von modulspezifischen Daten zuständig.

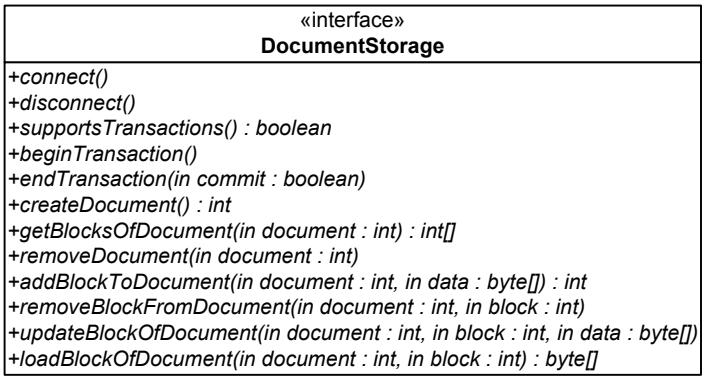

Abbildung 4.3: UML-Klassendiagramm der DocumentStorage-Schnittstelle des DMS, Quelle: Eigene Darstellung

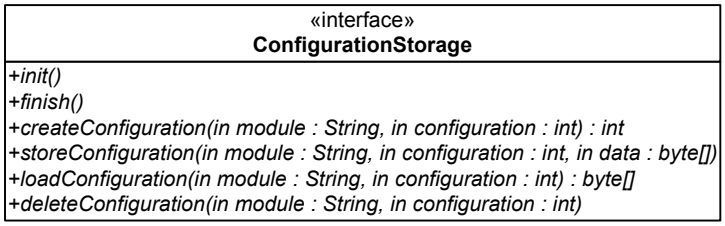

Abbildung 4.4: UML-Klassendiagramm der ConfigurationStorage-Schnittstelle, Quelle: Eigene Darstellung des DMS

Der Zugriff auf Dokumente über die Schnittstelle `DocumentStorage` erfolgt durch Instanziierung einer implementierenden Klasse. Jede Instanz kann nur eine Transaktion durchführen, werden mehrere gleichzeitig benötigt, so müssen entsprechend viele Instanzen erzeugt werden. Die Schnittstelle übernimmt nicht die Aufgaben der Verwaltung der Blöcke eines Dokuments. Dessen Struktur muss vom jeweiligen Modul selbst beziehungsweise über die zuständigen Klassen des Kernmoduls verwaltet werden.

Zusätzlich zu Dokumenten wird es jedem Modul gestattet, eigene Daten abzuspeichern. Zu diesem Zweck gibt es die Schnittstelle `ConfigurationStorage`. Im Unterschied zum `DocumentStorage` werden keine Transaktionen angeboten und die Konfigurationen Modulen direkt zugeordnet. Es unterliegt dem Modul selbst, welche Nummern die einzelnen Konfigurationen erhalten, da nur ein einzelnes Modul darauf zugreift und die gespeicherten Daten modulspezifisch sind.

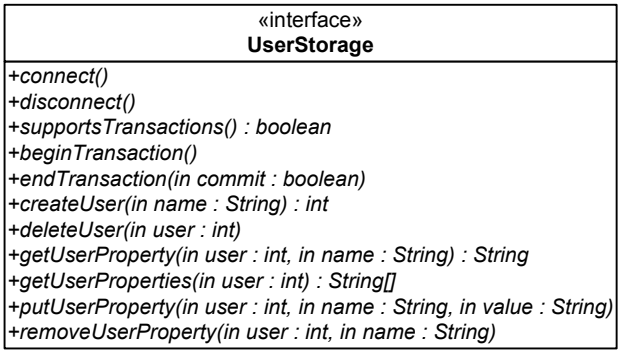

Abbildung 4.5: UML-Klassendiagramm der Modulschnittstelle für Benutzerspeicher des DMS, Quelle: Eigene Darstellung

4.2.2 Zugriff auf die Benutzerverwaltung

Der Zugriff auf die Benutzerverwaltung ermöglicht es Modulen benutzerspezifische Daten zu speichern und so Gruppenangehörigkeiten und Berechtigungen zu prüfen. Damit kann die Authorisierung des Zugriffs effizient umgesetzt werden.

Zur Benutzerverwaltung gehört einmal die Modulschnittstelle für die Speicherung der Daten `UserStorage`, die dazu genutzt werden soll, eine Anbindung an Verzeichnisdienste wie LDAP oder direkt an Datenbanksysteme zu realisieren, und zum anderem die Schnittstelle zum Zugriff auf die Benutzerrechte `UserManager`, die von den verschiedenen Modulen genutzt werden kann. Die UML-Klassendiagramme sind in Abbildung 4.5 beziehungsweise 4.6 dargestellt, die Quelltexte der Schnittstellen finden sich in den Anhängen C.3 und C.4.

Die Speicherung der Benutzerdaten funktioniert nahezu analog zur allgemeinen Datenspeicherung, nur dass keine beliebigen Datenblöcke gespeichert werden können, sondern nur Zeichenketten. Das Modul wird über die Konfigurationsdatei des Dispatchers angemeldet (Schlüssel ist `com.ewerk.dms.userstorage`).

Die Schnittstelle `UserManager` erlaubt das einfache Prüfen und Setzen von Benutzerrechten inklusive rekursiver Prüfung über die Gruppenrechte, aber auch das Management von Gruppenzugehörigkeiten. Ebenfalls können zur Ermittlung der Benutzer-IDs des die Anfrage ausführenden Benutzers Implementierungen der Schnittstelle verwendet werden. Ein Modul kann durch den Dispatcher eine Instanz der Klasse erhalten (siehe Anhang C.7).

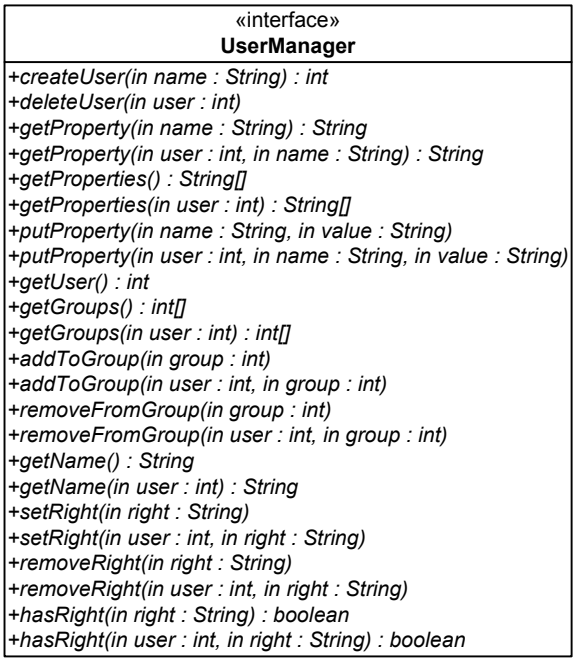

Abbildung 4.6: UML-Klassendiagramm des Benutzermanagements des DMS, Quelle: Eigene Darstellung

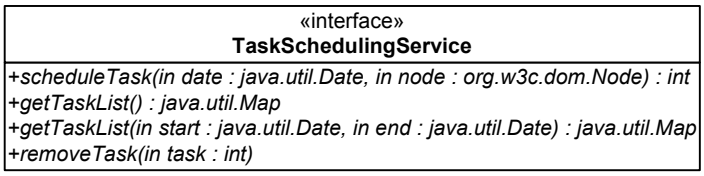

Abbildung 4.7: UML-Klassendiagramm des Task-Scheduling-Service des DMS, Quelle: Eigene Darstellung

4.2.3 Zugriff auf den Task-Scheduling-Service

Jedes Modul kann über den Dispatcher Zugriff auf den Task-Scheduling-Service (TSS, siehe Abschnitt 3.3.3) erhalten. Abbildung 4.7 zeigt den TSS als UML-Klassendiagramm, der Quelltext findet sich in Anhang C.8.

Der TSS bietet vier Methoden an:

scheduleTask Diese Funktion plant die Ausführung einer Anfrage an einem bestimmten Datum ein. Der zu übergebende DOM-Knoten muss der auszuführende Befehl sein, beispielsweise Zeilen 20 bis 23 in Abbildung 4.1.

```
1  // ...
2  TaskSchedulingService tss = Dispatcher.getInstance().getTaskSchedulingService();
3  tss.removeTask(oldTaskId);
4  int id = tss.scheduleTask(date, taskAsDOM);
5  // ...
```

Abbildung 4.8: Code-Fragment zum Zugriff auf den Task-Scheduling-Service

getTaskList Diese Funktion existiert mit zwei Signaturen. Einmal kann eine Liste aller noch auszuführenden Aufgaben ermittelt werden. Die zweite Variante lässt die Angabe eines Zeitintervalls zu, worin die Ausführung der Anfragen geplant sein soll.

removeTask Mit dieser Funktion lässt sich über die Identifikationsnummer der Aufgabe die geplante Ausführung löschen.

Der Zugriff auf den TSS erfolgt über den Dispatcher und ist im Code-Fragment in Abbildung 4.8 dargestellt.

4.2.4 Erweiterungsmodule

Das DMS ist so konzeptioniert, dass es fast beliebig erweitert werden kann. Um die Funktionalität zu vergrößern, können neue Module integriert werden, die andere Module erweitern oder neue Funktionen einführen können. Mit den weiter oben in diesem Abschnitt beschriebenen Schnittstellen kann jedes Erweiterungsmodul arbeiten und somit auf den Dokumentenbestand und auch auf eigene Daten bequem zugreifen, ohne die Konsistenz und Persistenz der Daten sichern zu müssen.

Erweiterungsmodule müssen entsprechend einer Java-Klassenschnittstelle implementiert und registriert werden und die Anfrage- und Antwortsprachen definieren. Weiter müssen Module über den Standard-Konstruktor korrekt instanziierbar sein, das heißt,

```
module = new Class.forName("Modulklassenname").newInstance();
```

muss eine funktionsfähige Instanz liefern.

4.2.4.1 Klassenschnittstelle

Damit ein Erweiterungsmodul in das System integriert werden kann, muss es die in Abbildung 4.9 dargestellte Schnittstelle (Interface) implementieren. Der Quelltext der Schnittstelle findet sich im Anhang C.5.

Abbildung 4.9: UML-Klassendiagramm der Modulschnittstelle des DMS, Quelle: Eigene Darstellung

Die in Abbildung 4.9 gezeigten Methoden teilen sich in Initialisierungs- bzw. Deinitialisierungsmethoden und die Verarbeitungsmethode auf. Die Methode `process` wird vom Dispatcher aufgerufen, falls das Modul für den Namensraum des Befehls registriert ist. Der zugehörige Namensraum wird als erster Parameter übergeben. Der zweite Parameter ist die Aufrufpriorität. Diese Priorität kann das Modul bei der Registrierung (siehe nächsten Unterabschnitt) frei wählen. Der dritte Parameter ist der DOM-Baum des Befehls und der vierte Parameter enthält alle Antworten der vorhergehenden Anfragen und alle nachfolgenden Elemente, also die Anfragen, die noch nicht beantwortet wurden.

Die in Abbildung 4.1 gezeigten Anfragen würde beispielsweise einen Aufruf des Groupware-Moduls erzeugen, der folgende Parameterwerte hätte:

namespaceURI http://www.ewerk.com/dms/groupware/request

priority 0

node DOM-Baum von Zeile 19 bis 29 Abbildung 4.1

allNodes Antworten der vorhergehenden Anfragen (Zeile 10 bis 16 aus dem Beispiel in Abbildung 4.2.

Die Parameter `node` und `allNodes` dürfen vom Modul verändert werden. Wird `node` verändert, dann wird der Dispatch-Vorgang für die Anforderung neu gestartet. So haben Module die Möglichkeit, eine Anfrage in eine neue umzuformen. Es können auch mehrere neue Anfragen in den Dispatcher gegeben werden, indem sie in die Liste `allNodes` an den entsprechenden Stellen eingefügt werden.

Die Rückgabe des Aufrufs von `process(...)` ist die an den Client zurückgelieferte Antwort und wird vom Dispatcher in ein `response`-Element eingebettet. Der zurückgelieferte DOM-Baum entspricht im Beispiel (Abbildung 4.2) den Zeilen 19 bis 28.

```
1  package somecompany.dms.extension;
2
3  import com.ewerk.dms.module.*;
4  import com.ewerk.dms.Dispatcher.*;
5
6  public class ExtensionModule implements DispatcherClient {
7          // ...
8          public void init() {
9                  // ...
10                 Dispatcher dispatcher = Dispatcher.getInstance();
11                 dispatcher.registerModule(this,
12                         "http://somecompanydomain/dms/extension/request");
13                 dispatcher.registerModule(this,
14                         "http://www.ewerk.com/dms/groupware/request", -1);
15                 dispatcher.registerModule(this,
16                         "http://www.ewerk.com/dms/groupware/request", 10);
17         }
18         // ...
19 }
```

Abbildung 4.10: Beispiel für eine Implementierung der Intialisierung eines Moduls

4.2.4.2 Registrierung des Moduls

Bei der Initialisierung des Dispatchers wird aus der Konfigurationsdatei des Dispatchers die Modulliste geladen (Schlüssel ist `com.ewerk.dms.dispatcher.modullist`). Nacheinander werden die gelisteten Module (genauer deren Klassen) instanziiert.

Im nächsten Schritt werden die Module vom Dispatcher durch den Aufruf der Methode `init()` initialisiert. Innerhalb dieser Methode muss sich ein Modul beim Dispatcher auf die von ihm verwendeten Namensräume registrieren. Die Abbildung 4.10 zeigt am Beispiel, wie dieser Schritt durchgeführt werden kann. Die Definition der Klasse Dispatcher findet sich zum besseren Verständnis im Anhang C.7.

Die Prioritäten (der dritte Parameter bei den Aufrufen von `registerModule()`) kann jedes Modul frei wählen. Sie bestimmen die Aufrufreihenfolge während der Abarbeitung des Dispatchers. Kleinere Werte führen dabei zu früherer Ausführung. Der Wertebereich des Datentyps `int` darf voll ausgeschöpft werden. Module dürfen sich – wie im Beispiel ersichtlich (Zeilen 16 bis 19) – mit unterschiedlichen Prioritäten mehrfach registrieren und werden dann auch mehrfach aufgerufen. Neue durch das Modul definierte Befehle sollten immer mit der Priorität 0 verarbeitet werden (Zeilen 14 und 15).

4.2.4.3 Definition der XML-Schemata

Jedes Modul muss zwei XML-Schemata definieren. Eines für die Anfrage-Sprache und eines für die Antwort darauf. Darüberhinaus darf das Modul auch weitere Schemata definieren.

Alle neuen Module müssen eigene Namensräume (Namespaces) verwenden. Die Namensräume haben den Präfix

`http://`*Firmendomain*`/dms/`*Modulname*`/`.

Anschließend werden die Namen der Sprachen angehangen (`request` für Abfragesprachen und `response` für Antwortsprachen).

In der Definition der Abfrage-Sprache muss für jeden Befehl ein Element definiert werden. Jedes dieser Elemente muss neben den eigenen Parametern Platz für Erweiterungen lassen und ein optionales Element `extensions` am Ende des Elementinhalts zulassen, dass beliebige Elemente beliebiger Namensräume enthalten kann. Im Anhang B.1 finden sich in den Definitionen der Kernmodule Beispiele hierfür.

Die Definition der Antwortsprachen geschieht nahezu analog, nur dass die Inhalte der Befehlselemente die Ausgabedaten sind.

4.2.5 Erweiterungsmodule für die Authentifizierung

Erweiterungsmodule, die neue Authentifizierungsmethoden bereitstellen, müssen zusätzlich zu den in Abschnitt 4.2.4 genannten Bedingungen die Schnittstelle `AuthenticationModule` implementieren. Diese Schnittstelle ist in Abbildung 4.11 als UML-Klassen-Diagramm dargestellt, der Quelltext befindet sich im Anhang C.6. Sie enthält nur eine Methode (`authenticate`). Diese Methode wird vom Dispatcher als erster Schritt vor allen anderen Anfragebearbeitungen aufgerufen. Der übergebene Parameter `node` beinhaltet den DOM-Baum zur Authentifizierung. Im Beispiel aus Abbildung 4.1 entspricht das den Zeilen 6 bis 18. Der Rückgabewert schließlich gibt an, ob eine Authentifizierung erfolgreich war. Zu beachten gilt es, dass Module, die die übergebene Anfrage nicht abarbeiten können, immer den Wert `false` zurückliefern, da die Anfragen an alle registrierten Module weitergegeben werden.

Eine weitere Forderung an Erweiterungsmodule zur Authentifizierung ist, dass sie stets im Namensraum *http://www.ewerk.com/dms/auth/request* arbeiten und die darin ent-

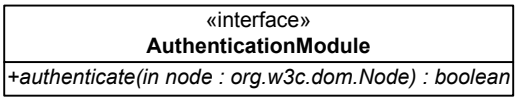

Abbildung 4.11: UML-Klassendiagramm der Modulschnittstelle zur Authentifizierung, Quelle: Eigene Darstellung

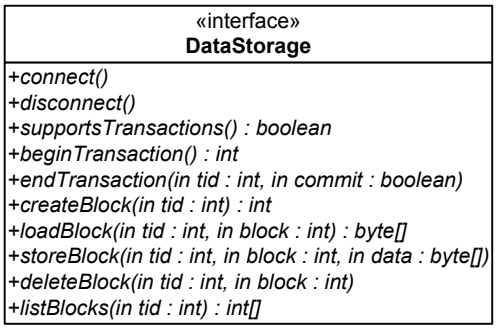

Abbildung 4.12: UML-Klassendiagramm der Modulschnittstelle für Datenspeicher, Quelle: Eigene Darstellung

haltenen Abfrage `get_auth_method_list` erweitern. Das XML-Schema zum Format dieser Liste findet sich in Anhang B.1.3.

Die Anmeldung des Moduls am Dispatcher geschieht analog zu der von normalen Erweiterungsmodulen (siehe Abschnitt 4.2.4.2). Zusätzlich muss das Modul die Methode `registerAuthenticationModule()` des Dispatchers aufrufen.

4.2.6 Erweiterungsmodule für Datenspeicher

Im Gegensatz zu den bereits oben beschriebenen Erweiterungsmodulen bearbeiten jene für Datenspeicher keine direkten Anfragen, sondern implementieren die Schnittstelle zur Anbindung von Speichersystemen, die dann direkt von den in Abschnitt 4.2.1 und 4.2.2 vorgestellten Klassen verwendet werden können.

Das DMS kann stets nur ein Modul zum Zugriff auf den Datenspeicher verwenden, das heißt, wenn dieses Modul ausgetauscht wird, sind die vorher gespeicherten Daten nicht mehr verfügbar und müssen gegebenenfalls auf das neue System dupliziert werden.

Erweiterungsmodule für Datenspeicher müssen Funktionen zum Verwalten von Binärdatenblöcken unter bestimmten ganzzahligen Schlüsselwerten bereitstellen. Dazu müssen sie die in Abbildung 4.12 gezeigte Schnittstelle implementieren; der Quelltext findet sich im Anhang C.9. Diese Schnittstelle fordert folgende Methoden:

connect Diese Methode wird während des Systemstarts aufgerufen. Sie soll das Modul initialisieren und die Verwendung vorbereiten.

disconnect Diese Methode wird kurz vor dem Herunterfahren des Systems aufgerufen und muss dafür sorgen, dass alle noch vorhandenen Transaktionen mit einem Rollback abgeschlossen werden.

supportsTransactions Diese Methode soll angeben, ob das Modul Transaktionen unterstützt oder nicht.

beginTransaction Die Methode `beginTransaction` startet eine neue Transaktion. Sie liefert entsprechend eine Transaktions-ID zurück, worüber auf die Transaktion zugegriffen werden kann. Unterstützt das Modul keine Transaktionen, muss es -1 zurückliefern und darf in anderen Methoden die Transaktions-IDs ignorieren.

endTransaction Beendet eine Transaktion. Ist der Parameter `commit` vom Wert `true`, sollen alle Änderungen gespeichert werden, ansonsten ist ein Rollback durchzuführen.

createBlock Diese Methode erstellt einen neuen Block und liefert einen Primärschlüssel zum Block zurück.

loadBlock Diese Methode lädt einen bestimmten Block in einen Speicherbereich.

storeBlock Die Methode `storeBlock` sichert einen Speicherbereich in einen vorhandenen Block.

deleteBlock Diese Methode löscht einen Block aus dem Datenspeicher.

listBlocks Diese Methode muss eine Liste aller verwendeten Primärschlüssel zurückliefern.

Zur Verwendung des Moduls muss der Schlüssel `com.ewerk.dms.storage.datastorage` in der Dispatcher-Konfigurationsdatei auf den Klassennamen des Moduls gesetzt werden.

4.2.7 Erweiterungsmodule für Dokumentenein- und -ausgabe

Der Datenaustausch mit dem Benutzer kann, wie bereits in Abschnitt 2.4.4 erläutert, auf viele verschiedene Arten durchgeführt werden. Um bezüglich der Methoden des Datenaustauschs flexibel zu bleiben, besitzt das Dokumentenmanagementsystem eine allgemeine Schnittstelle zu Erweiterungsmodulen für diesen Zweck.

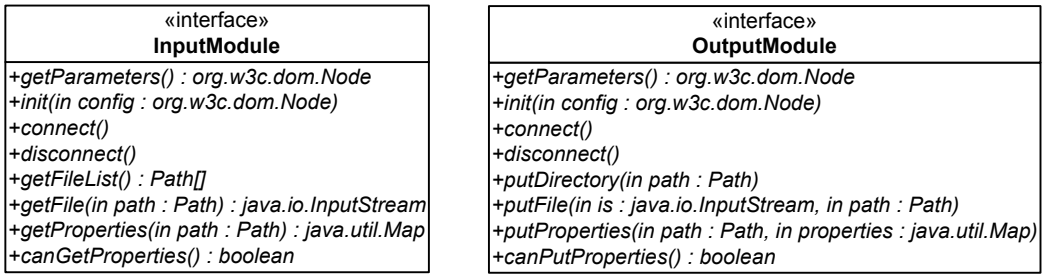

Abbildung 4.13: UML-Klassendiagramm der Modulschnittstelle für Dokumenteneinund -ausgabe, Quelle: Eigene Darstellung

Grundsätzlich kann zwischen Ein- und Ausgabemodulen unterschieden werden, da nicht alle Möglichkeiten zum Datenaustausch beide Richtungen unterstützen, wie beispielsweise das Herunterladen mit Internet-Browser. In Abbildung 4.13 sind diese Schnittstellen dargestellt, die Quelltexte befinden sich in den Anhängen C.10 und C.11.

Folgende Methoden müssen durch Eingabemodule implementiert werden:

getParameters Diese Methode liefert eine Beschreibung der Parameter, die zum Aufbau einer Verbindung zum Benutzerkonto benötigt werden. Dies sind normalerweise Konten-Namen und Passwörter. Die Übergabe erfolgt als DOM-Teilbaum, der gemäß den in Anhang B.1.5 definierten Strukturen aufgebaut sein muss.

init Diese Methode wird zum Intialisisieren vor der Verwendung mit den Parametern des Moduls aufgerufen.

connect Diese Methode wird vor der Verwendung aufgerufen. Erst danach werden die Methoden zum Zugriff auf die Daten verwendet.

disconnect Diese Methode wird nach der Verwendung aufgerufen. Danach werden die Methoden zum Zugriff auf die Daten nicht mehr verwendet.

getFileList Diese Methode ermittelt die Liste der Eingabedateien und -pfade des Dokuments und liefert sie als Feld zurück.

getFile Diese Methode liefert einen Datenstrom zum Lesen einer bestimmten Datei.

getProperties Diese Methode liefert die Metadaten eines Dokuments oder `null` zurück.

canGetProperties Diese Methode gibt an, ob das Modul den Transfer von Metadaten zulässt oder nicht.

Für Ausgabemodule sind folgende Funktionen zu implementieren:

getParameters, init, connect, disconnect Diese Methoden haben zu den oben erwähnten analoge Bedeutung.

putDirectory Diese Methode legt ein Verzeichnis an.

putFile Diese Methode speichert eine Datei im Ausgabekonto.

putProperties Diese Methode speichert die Metadaten zu einer Datei oder einem Verzeichnis ab, falls dies unterstützt wird.

canPutProperties Die Methode gibt an, ob es möglich ist, Metadaten zu transferieren.

Zur Registrierung der Module müssen diese in der Konfigurationsdatei des Dispatchers eingetragen werden (Schlüssel `com.ewerk.dms.dispatcher.inputmodules` und `com.ewerk.dms.dispatcher.outputmodules`).

5 Ausblick

Der Vorschlag für dieses Dokumentenmanagementsystem lässt einige wichtige Punkte offen. Der Entwurf der Bereiche der Suche im Dokumentenbestand und des Workflowmanagements ist noch nicht erfolgt, muss aber direkt anschließen, da es sich um unabdingbare Funktionen handelt, die über den Erfolg des Systems entscheiden. Hier ist vor allem die Anbindung an das schon vorhandene Contentmanagementsystem conrad zu beachten, dass ähnliche Funktionen bereits aufweist.

Im Anschluss ist auf jeden Fall noch der Großteil der Implementierungsarbeit zu leisten. Insbesondere die vorgeschlagenen Techniken XML und Enterprise Java Beans müssen auf die notwendige Performanz geprüft werden. Wichtige Punkte bei diesem Test sollte der Einfluss der verwendeten CTMs, der Bedarf an Arbeitspeicher und die Reaktionszeiten des System bei verschieden starker Auslastung sein. Da das DMS hinsichtlich Performanz entworfen wurde, wird sicherlich diesen Anforderungen gerecht werden.

Die erste funktionsfähige Implementierung sollten in einem kleinen Umfeld getestet werden, um die sicherlich vorhandenen Kinderkrankheiten des Systems zu beseitigen. Dazu ist aber auch die Schaffung des Web-Clients notwendig, der noch vollständig zu entwickeln ist. Weiter sollte intensiv an der Administrierung des Systems gearbeitet werden. Ziel muss es sein, eine einfache und flexible Schnittstelle für die administrierenden Personen zu schaffen. Dann wäre ein Testumfeld innerhalb der eWerk IT GmbH denkbar, welches gut kleine kooperative Umgebungen repräsentieren kann.

Erst wenn diese Feuerproben bestanden sind, kann das System in den Arbeitsbetrieb übergehen und es wird dann mit Sicherheit anders aussehen, als es jetzt absehbar ist.

Die Chancen für ein DMS mit umfassender Unterstützung von kooperativer Arbeit sehen gut aus, da der Markt bisher noch nicht gut abgedeckt ist. Die sich zur Zeit verstärkende Globalisierung wird sicher neue Einsatzgebiete für kooperationsunterstützende Software eröffnen und Entwicklungen im Gebiet des Dokumentenmanagements verstärken.

Literaturverzeichnis

[1] Kap. Managing Large Collections of Documents In: Bannan, Joan: *Intranet Document Management: A Guide for Webmasters and Content Providers*. Addison-Wesley, April 1997. – ISBN 0–20187–379–6

[2] Borghoff, Uwe M. ; Schlichter, Johann H.: *Computer-Supported Cooperative Work*. Springer, 2000. – ISBN 3–540–66984–1

[3] Bentley, R. ; Appelt, W. ; Busbach, U. ; Hinrichs, E. ; Kerr, D. ; Sikkel, K. ; Trevor, J. ; Woetzel, G.: Basic Support for Cooperative Work on the World Wide Web. In: *International Journal of Human Computer Studies* 46 (1997), Juni, Nr. 6, S. 827–846

[4] Information Sciences Institute, University of Southern California. *Transport Control Protocol, RFC 793*. URL http://www.faqs.org/rfcs/rfc793.html Besucht am 15. Juli 2003. September 1981

[5] Information Sciences Institute, University of Southern California. *Internet Protocol, RFC 791*. URL http://www.faqs.org/rfcs/rfc791.html Besucht am 15. Juli 2003. September 1981

[6] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Berners-Lee, T. *Hypertext Transfer Protocol – HTTP/1.1., RFC 2068*. URL http://www.ietf.org/rfc-/rfc2068.txt. January 1997

[7] Intranet Journal. *Intranet FAQ: Intranet Basics – 1. Intranet defined*. URL http://www.intranetjournal.com/faqs/whatis/1.html Besucht am 15. Juli 2003

[8] Saffady, William. *The Document Life Cycle: A White Paper*. URL http://www.documentconversion.com/news2.htm Besucht am 20. Oktober 2002

[9] *UStG 1999 (Umsatzsteuergesetz 1999, idF v. 19.12.2001), §14a Abs 5.* URL http://www.steuernetz.de/gesetze/ustg/20011219/volltext.html Besucht am 15. April 2003

[10] *BDSG (Bundesdatenschutzgesetz).* URL http://www.netlaw.de/gesetze/bdsg.htm Besucht am 15. April 2003

[11] *TDSV (Telekommunikationsdienstunternehmen-Datenschutzverordnung), §7 Abs. 3.* URL http://www.netlaw.de/gesetze/tdsv.htm Besucht am 15. April 2003

[12] Initiative, Dublin Core M. *Dublin Core Metadata Initiative (DCMI) Homepage.* URL http://dublincore.org Besucht am 18. Januar 2003

[13] Burger, Franz ; Reich, Siegfried: Usability of Groupware Products for Supporting Publishing Workflows. In: *Proceedings of Connectivity 1994. Ninth Austrian-Hungarian Informatics Conference*, 1994. – URL ftp://ftp.ifs.uni-linz.ac.at/pub-/publications/1994/2194.pdf. – ISBN 3–85403–076–2, S. 51–64

[14] Burger, Franz ; Reich, Siegfried: Entwurf und Implementierung eines integrierten Dokumenten- und Workflowmanagement-Systems für elektronisches Publizieren. In: *Wirtschaftsinformatik* 37 (1996), Nr. 2, S. 191–198. – ISSN 0937–6429

[15] Price, Derek ; CollabNet [u. a.]. *Concurrent Versions System (CVS) Website.* URL http://www.cvshome.org Besucht am 20. April 2003

[16] *Source Code Control System (SCCS).* URL http://cssc.sourceforge.net/old-cyclic-/sccs.html Besucht am 20. April 2003

[17] *Revision Control System (RCS) Official Homepage.* URL http://www.cs.purdue.edu/homes/trinkle/RCS Besucht am 20. April 2003

[18] Free Software Foundation, Inc. *GNU diffutils.* URL http://www.gnu.org-/software/diffutils/diffutils.html Besucht am 20. April 2003

[19] Free Software Foundation, Inc. *GNU patch.* URL http://www.gnu.org-/software/patch/patch.html Besucht am 20. April 2003

[20] MacDonald, Joshua P. *Xdelta.* URL http://www.xcf.berkeley.edu/∼jmacd-/xdelta.html Besucht am 20. April 2003

[21] MacDonald, Joshua P. *File System Support for Delta Compression*

[22] Universitätsbibliothek Regensbur. *Regensburger Verbundklassifikation (RVK).* URL http://www.bibliothek.uni-regensburg.de/Systematik-/systemat.html Besucht am 21. April 2003

[23] Yahoo! Inc. *Yahoo!* URL http://www.yahoo.com Besucht am 22. April 2003

[24] WEB.DE AG. *WEB.DE.* URL http://www.web.de Besucht am 22. April 2003

[25] Kap. Indexieren, Klassieren, Extrahieren In: Knorz, Gerhard: *Grundlagen der praktischen Information und Dokumentation.* Saur Verlag, 1996, S. 120–140. – ISBN 3–598–11309–9

[26] Breure, L. *Development of the Genre Concept.* URL http://www.cs.uu.nl-/people/leen/GenreDev/GenreDevelopment.htm Besucht am 24. April 2003. 2001

[27] Honkaranta, A.: Developing Document and Content Management in Enterprises Using a „Genre Lens“. In: *Proceedings of the 5th International Conference on Enterprises Information Systems*, 2003. – URL http://www.cc.jyu.fi/ ankarjal-/ICEIS2003_GenreDM.pdf und http://www.cc.jyu.fi/ ankarjal/ICEIS2003app.pdf

[28] Chandler, D. *An introduction to genre theory.* URL http://www.aber.ac.uk-/media/Documents/intgenre/intgenre.html Besucht am 24. April 2003

[29] Tyrväinen, Pasi ; Päivärinta, Tero: On Rethinking Organizational Document Genres for Electronic Document Management. In: *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Institute of Electrical and Electronics Engineers (IEEE), 1999

[30] Karjalainen, Anne ; Päivärinta, Tero ; Tyrväinen, Pasi ; Rajala, Jari: Genre-Based Metadata for Enterprise Document Management. In: *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Institute of Electrical and Electronics Engineers (IEEE), 2000

[31] Frakes, William ; Baeza-Yates, Ricardo: *Infomation Retrieval: Data Structures and Algorithms.* Prentice Hall, 1993

[32] *Google.* URL http://www.google.com Besucht am 24. April 2003

[33] *AltaVista.* URL http://www.altavista.com Besucht am 24. April 2003

[34] NEC Research Institute, Inc. *CiteSeer. Scientific Literature Digital Library.* URL http://www.citeseer.com Besucht am 24. April 2003

[35] Adaptive READ. *Adaptive READ – Lesen und Lernen – vom Dokument zum Wissen.* URL http://adaptive-read.de/aktuell/pdf/info.pdf

[36] Adaptive READ. *Befragung im Rahmen des Projekts Adaptive READ: Kurzanalyse der Umfrageergebnisse der 1. Befragung.* URL http://www.adaptive-read.de-/aktuell/pdf/Analyse.pdf Besucht am 28. Oktober 2002

[37] Adaptive READ. *Befragung im Rahmen des Projekts Adaptive READ: Ergebnisse der 1. Umfrage - ein Überblick.* URL http://www.adaptive-read.de/aktuell-/pdf/Ueberblick.pdf Besucht am 28. Oktober 2002

[38] *Telearbeit – Leitfaden für flexibles Arbeiten in der Praxis.* Bundesministerium für Wirtschaft und Technologie; URL http://www.bmwi.de/Homepage/download-/Arbeit/Telearbeit.pdf. Februar 2001

[39] empirica Gesellschaft für Kommunikations-und Technikforschung. *Benchmarking Telework in Europe 1999 – Auswertung des General Population Survey.* URL http://www.ecatt.com/statistics/tgps/gpsdatenreport.pdf Besucht am 25. April 2003

[40] IBM Corporation. *IBM Lotus software.* URL http://www.lotus.com Besucht am 21. April 2003

[41] Bentley, Richard ; Horstmann, Thilo ; Trevor, Jonathan: The World Wide Web as enabling technology for CSCW: The case of BSCW. In: *Computer-Supported Cooperative Work* 6 (1997), Nr. 2, S. 111–134. – URL http://bscw.gmd.de/Papers/CSCWJ-WWW/CSCW_journal.ps. – ISSN 0925–9724

[42] Cingar-Golowko, N. *Integration von Dokumentenmanagement-Systemen mit Workflow- und Groupware-Systemen.* August 2000

[43] Appelt, Wolfgang: What Groupware Functionality Do Users Really Use? Analysis of the Usage of the BSCW System. In: *Proceedings of the Ninth Euromicro Workshop on Parallel and Distributed Processing (PDP '01)*, IEEE Computer Society, 2001. – URL http://bscw.gmd.de/Papers/PDP2001/PDP2001.pdf

[44] Brush, Alice Jane B.: Annotating Digital Documents for Asynchronous Collaboration / Department of Computer Science and Engineering, University of Washington. 2002 (02-09-02). – Forschungsbericht. URL ftp://ftp.cs.washington.edu-/tr/2002/09/UW-CSE-02-09-02.pdf

[45] Microsoft Corporation. *Microsoft Office*. URL http://www.microsoft.com-/germany/office/ Besucht am 3. April 2003

[46] Adobe Systems Incorporated. *Adobe Acrobat*. URL http://www.adobe.de-/products/acrobat/ Besucht am 3. April 2003

[47] Educational Technology Development Group, The University of Washington. *EPost. UW Catalyst Toolkit*. URL http://catalyst.washington.edu-/tools/ Besucht am 3. April 2003

[48] Moraes, Mark. *What is Usenet?* URL http://www.faqs.org/faqs/usenet/what-is/part1/ Besucht am 3. April 2003

[49] Georgakopoulos, Dimitrios ; Hornick, Mark F. ; Sheth, Amit P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: *Distributed and Parallel Databases* 3 (1995), Nr. 2, S. 119–153. – URL ftp://ftp.gte.com/pub/dom/reports/GEOR95a.ps

[50] Abbott, K. R. ; Sarin, S. K.: Experiences with workflow management: Issues for the next generation. In: Cizmar, D. (Hrsg.): *Proceedings of the ACM Conference on Computer- Supported Cooperative Work (CSCW '94)*. Chapel Hill, North Carolina : ACM Press, 1994. – URL http://faculty.cis.drexel.edu/class/info616-/Readings/Paper13.pdf Besucht am 3. April 2003, S. 113–120

[51] Rouncefield, Mark ; Hughes, John A. ; Rodden, Tom ; Viller, Stephen: Working with „constant interruption“: CSCW and the small office. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ACM Press, 1994. – ISBN 0–89791–689–1, S. 275–286

[52] Internet Engineering Task Force / Internet Society. *IETF LDAP-bis Working Group*. URL http://www.ietf.org/html.charters/ldapbis-charter.html Besucht am 21. April 2003

[53] Sun Microsystems, Inc. *NFS: Network File System Protocol Specification, RFC 1094*. URL http://www.ietf.org/rfc/rfc1094.txt

[54] Sharpe, Richard. *What is SMB?* URL http://www.samba.org/cifs/docs/what-is-smb.html Besucht am 21. April 2003

[55] Postel, J. ; Reynolds, J. *File Transfer Protocol, RFC 959*. URL http://www.ietf.org/rfc/rfc959.txt

[56] Internet Engineering Task Force / Internet Society. *IETF Secure Shell (secsh) Working Group*. URL http://www.ietf.org/html.charters/secsh-charter.html Besucht am 21. April 2003

[57] OpenBSD. *OpenSSH Website*. URL http://www.openssh.org Besucht am 21. April 2003

[58] E. James Whitehead, Jr. ; Goland, Yaron: WebDAV: a network protocol for remote collaborative authoring on the Web. In: *Proceedings of the Sixth European conference on Computer supported cooperative work*, Kluwer Academic Publishers, 1999. – URL http://www.ics.uci.edu/ ejw/papers/dav-ecscw.pdf. – ISBN 0–7923–5948–8, S. 291–310

[59] E. James Whitehead, Jr.: World Wide Web distributed authoring and versioning (WebDAV): an introduction. In: *StandardView* 5 (1997), Nr. 1, S. 3–8. – URL http://www.ics.uci.edu/pub/ietf/webdav/intro/dav_acm_sv.pdf. – ISSN 1067–9936

[60] Goland, Y. ; Whitehead, J. ; Faizi, A. ; Carter, S. ; Jensen, D. *HTTP Extensions for Distributed Authoring - WebDAV, RFC 2518*. URL ftp://ftp.isi.edu-/in-notes/rfc2518.txt. Februar 1999

[61] Stein, Greg. *WebDAV Website*. URL http://www.webdav.org Besucht am 19. April 2003

[62] Clemm, G. ; Amsden, J. ; Ellison, T. ; Whitehead, J. *Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning), RFC 3253*. URL http://www.ietf.org/rfc/rfc3253.txt. March 2002

[63] Whitehead, Jim. *DeltaV: Adding Versioning to the Web, WWW10 Tutorial Notes*. URL http://www.webdav.org/deltav/WWW10/deltav-intro.pdf Besucht am 19. April 2003

[64] Clemm, G. ; Hopkins, A. ; Sedlar, E. ; Whitehead, J. *WebDAV Access Control Protocol*. URL http://www.webdav.org/acl/protocol/draft-ietf-webdav-acl-09.pdf Besucht am 19. April 2003

[65] Slein, J. ; Jr., E.J. W. ; Davis, J. ; Clemm, G. ; Fay, C. ; Crawford, J. *WebDAV Ordered Collections Protocol*. URL http://www.ietf.org/internet-drafts-/draft-ietf-webdav-ordering-protocol-07.txt

[66] Slein, J. ; Jr., E.J. W. ; Davis, J. ; Clemm, G. ; Fay, C. ; Crawford, J. *WebDAV Redirect Reference Resources*. URL http://www.cs-ipv6.lancs.ac.uk-/ipv6/documents/standards/general-comms/internet-drafts/draft-ietf-webdav-redirectref-protocol-03.txt

[67] Qu, Changtao ; Nejdl, Wolfgang: Constructing a web-based asynchronous and synchronous collaboration environment using WebDAV and Lotus SameTime. In: *Proceedings of the 29th annual ACM SIGUCCS conference on User services*, ACM Press, 2001. – URL http://www.kbs.uni-hannover.de/ changtao/016Qu.pdf. – ISBN 1–58113–382–0, S. 142–149

[68] Qu, Changtao ; Engel, Thomas ; Meinel, Christoph: Implementation of an Enterprise-Level Groupware System Based on J2EE Platform and WebDAV Protocol. In: *Proceedings of the Fourth International Enterprise Distributed Object Computing Conference (EDOC'00)*, IEEE Computer Society, 2000. – URL http://www.kbs.uni-hannover.de/ changtao/Edoc.pdf

[69] Qu, Changtao ; Engel, Thomas ; Meinel, Christoph: Implementation of a Document Management System based on WebDAV Protocol. In: *Proceedings of the 1st International Conference on Management of Innovation and Technology (IEEE ICMIT 2000)*, IEEE Computer Society, November 2000. – URL http://www.kbs.uni-hannover.de/ changtao/Icmit.pdf

[70] Hinrichs, Joachim ; Nagler, Brigitte. *Archivierung und Dokumenten- Management als aktiver Gestaltungsprozess*. URL http://www.prowis.informatik.uni-bremen.de/pub/archiv.pdf Besucht am 31. Oktober 2002

[71] Blythin, Steve ; Hughes, John A. ; Kristoffersen, Steinar ; Rodden, Tom ; Rouncefield, Mark: Recognising 'Success' and 'Failure': Evaluating Groupware in a Commercial Context / Lancaster University, Computing Department, Cooperative Systems Engineering Group. 1997 (CSEG/25/1997). – Forschungsbericht. URL ftp://ftp.comp.lancs.ac.uk/pub/reports/1997/CSEG.25.97.pdf

[72] Gauss Interprise AG. *Gauss VIP DocManager*. URL http://www.gauss.de/gaussvip_p/gaussvip/EMEA/DE/PRODUKTE-/VIPEnterprise/VIPDocManager/index.jsp Besucht am 22. April 2003

[73] FileNet Corporation. *FileNet Content Manager*. URL http://www.filenet.com/English/Products/Content_Manager/ Besucht am 22. April 2003

[74] Comma Soft AG. *Infonea*. URL http://www.infonea.com/produkt/pro-ueb.htm Besucht am 22. April 2003

[75] Oracle Corporation. *Oracle Files*. URL http://otn.oracle.com/products-/ofiles/content.html Besucht am 22. April 2003

[76] Xerox Corporation. *DocuShare*. URL http://docushare.xerox.com Besucht am 22. April 2003

[77] Documentum, Inc. *Documentum Enterprise Document Management*. URL http://www.documentum.com/solutions/edm/edm_home.htm Besucht am 22. April 2003

[78] Hummingbird Ltd. *Hummingbird DM*. URL http://www.hummingbird.com/products/dm/overview.html Besucht am 22. April 2003

[79] Hyperwave AG. *Hyperwave Document Management*. URL http://www.hyperwave.com/d/solutions/horizontal/dm/

[80] AG, IXOS S. *IXOS eCONserver*. URL http://www.ixos.com/en/solutions/suite-/econserver/index.html Besucht am 22. April 2003

[81] Open Text Corporation. *OpenText LiveLink*. URL http://www.opentext.com-/livelink/applications.asp?id=1032 Besucht am 22. April 2003

[82] IBM Corporation. *Lotus Domino.Doc*. URL http://www.lotus.com-/dominodoc/ Besucht am 22. April 2003

[83] SAPERION AG. *Saperion Dokumenten-Server*. URL http://www.saperion.com Besucht am 22. April 2003

[84] DocuWare AG. *Docuware Document Server*. URL http://www.docuware.com Besucht am 22. April 2003

[85] Group, PostgreSQL Global D. *PostgreSQL Website*. URL http://www.postgresql.org Besucht am 22. April 2003

[86] Microsoft Corporation. *Microsoft Internet Explorer Technology Overview*. URL http://www.microsoft.com/windows/ie/evaluation/overview/default.asp Besucht am 24. April 2003

[87] Netcape. *Communicator Client Archive*. URL http://wp.netscape.com/-download/archive/client_archive47x.html Besucht am 24. April 2003

[88] Opera Software ASA. *Opera Product Overview*. URL http://www.opera.com-/products/ Besucht am 24. April 2003

[89] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Masinter, L. ; Leach, P. ; Berners-Lee, T. *Hypertext Transfer Protocol, RFC 2616*. URL http://www.ietf.org/rfc/rfc2616.txt Besucht am 24. April 2003. June 1999

[90] Freier, Alan O. ; Karlton, Philip ; Kocher, Paul C. *The SSL Protocol Version 3.0*. URL http://wp.netscape.com/eng/ssl3/draft302.txt Besucht am 24. April 2003. November 1996

[91] Dierk, T. ; Allen, C. *TLS Protocol Version 1.0, RFC 2246*. URL http://www.ietf.org/rfc/rfc2246.txt Besucht am 24. April 2003. January 1999

[92] Sun Microsystems, Inc. *Java Website*. URL http://java.sun.com Besucht am 24. April 2003

[93] Lindholm, Tim ; Yellin, Frank: *The Java Virtual Machine Specification*. Second Edition. Addison-Wesley, 1999. – URL http://java.sun.com/docs/books/vmspec-/index.html Besucht am 2. Mai 2003. – ISBN 0–201–43294–3

[94] Sun Microsystems, Inc. *Java Software FAQ*. URL http://java.sun.com/getjava-/faq.html#faq4 Besucht am 2. Mai 2003

[95] Sun Microsystems, Inc. *Java 2 Platform, Enterprise Edition (J2EE)*. URL http://java.sun.com/j2ee Besucht am 2. Mai 2003

[96] Sun Microsystems, Inc. *Java API Documentation*. URL http://java.sun.com-/api Besucht am 2. Mai 2003

[97] Sun Microsystems, Inc. *Java Technologies*. URL http://java.sun.com/products Besucht am 2. Mai 2003

[98] Sun Microsystems, Inc. *Java Legal and Licensing FAQ*. URL http://servlet.java.sun.com/help/legal_and_licensing Besucht am 2. Mai 2003

[99] Sun Microsystems, Inc. *Java 2 Platform Enterprise Edition Specification, v1.4*. URL http://java.sun.com/j2ee/j2ee-1_4-pfd3-spec.pdf Besucht am 2. Mai 2003

[100] The Middleware Company. *TheServerSide Application Server Matrix*. URL http://www.theserverside.com/reviews/matrix.jsp Besucht am 2. Mai 2003

[101] Sun Microsystems, Inc. *Java Servlet Specification Version 2.3*. URL http://www.jcp.org/aboutJava/communityprocess/final/jsr053 Besucht am 2. Mai 2003

[102] Sun Microsystems, Inc. *Java Servlet Servers and Engines*. URL http://java.sun.com/products/servlet/industry.html Besucht am 2. Mai 2003

[103] Apache Software Foundation, Inc. *Apache Tomcat*. URL http://jakarta.apache.org/tomcat Besucht am 2. Mai 2003

[104] Monson-Haefel, Richard: *Enterprise Java Beans*. O'Riley, 2001. – Deutsche Ausgabe der 2. Auflage. – ISBN 3–89721–192–0

[105] Sun Microsystems, Inc. *Enterprise JavaBeans Specification 2.1 Proposed Final Draft*. URL http://java.sun.com/products/ejb/docs.html Besucht am 20. Mai 2003. August 2002

[106] The JBoss Group, LLC. *JBoss*. URL http://www.jboss.org Besucht am 24. Juni 2003

[107] Sun Microsystems, Inc. *The JDBC API Universal Data Access for the Enterprise*. URL http://java.sun.com/products/jdbc/overview.html Besucht am 2. Mai 2003

[108] Applied Information Technologies, Inc. *SQL.ORG Website*. URL http://www.sql.org Besucht am 2. Mai 2003

[109] Sun Microsystems, Inc. *JDBC Drivers*. URL http://industry.java.sun.com-/products/jdbc/drivers Besucht am 2. Mai 2003

[110] Bray, Tim (Hrsg.) ; Paoli, Jean (Hrsg.) ; Sperberg-McQueen, C. M. (Hrsg.) ; Maler, Eve (Hrsg.). *Extensible Markup Language (XML) 1.0*. URL http://www.w3.org/TR/2000/REC-xml-20001006.pdf Besucht am 19. Mai 2003. Oktober 2000

[111] OASIS. *XML.org Website*. URL http://www.xml.org Besucht am 19. Mai 2003

[112] Fallside, David C. (Hrsg.). *XML Schema Part 0: Primer*. URL http://www.w3.org/TR/xmlschema-0 Besucht am 19. Mai 2003. Mai 2001

[113] The Apache Software Foundation. *Xerces2 Java Parser*. URL http://xml.apache.org/xerces2-j Besucht am 19. Mai 2003

[114] Bray, Tim (Hrsg.) ; Hollander, Dave (Hrsg.) ; Layman, Andrew (Hrsg.). *Namespaces in XML*. URL http://www.w3.org/TR/REC-xml-names Besucht am 19. Mai 2003. Januar 1999

[115] World Wide Web Consortium (W3C). *World Wide Web Consortium Website*. URL http://www.w3.org Besucht am 19. Mai 2003

[116] W3C DOM Working Group. *Document Object Model (DOM)*. URL http://www.w3.org/DOM Besucht am 20. Mai 2003

[117] Syverson, Paul F. ; Cervesato, Iliano: The Logic of Authentication Protocols. In: Focardi, R. (Hrsg.) ; Gorrieri, R. (Hrsg.): *Foundations of Security Analysis and Design*. Springer-Verlag LNCS 2171, 2001. – URL http://theory.stanford.edu/∼iliano/papers/fosad00.pdf.gz. – ISBN 3–540–42896–8

[118] Laboratories, RSA. *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1*. URL http://www.rsasecurity.com/rsalabs/faq/files/rsalabs_faq41.pdf Besucht am 27. Mai 2003. 2000

[119] Sandhu, Ravi S. ; Coyne, Edward J. ; Feinstein, Hal L. ; Youman, Charles E.: Role-Based Access Control Models. In: *IEEE Computer* 29 (1996), Nr. 2, S. 38–47. – URL http://citeseer.nj.nec.com/sandhu96rolebased.html

A Funktionsübersicht

In diesem Anhang befinden sich die Definitionen der Funktionen der einzelnen Module des DMS, die Client-Anwendungen und anderen Modulen bereitgestellt werden. Welche Funktionen den Clientanwendungen bereitgestellt werden, wird in Abschnitt 4.1.1 ausführlich dargestellt.

A.1 Kernmodul des Dokumentenmanagements

Im folgenden wird eine Übersicht über die Funktionen des Moduls gegeben. Sie entspricht im größtenteils den Funktionen, auf die Benutzer über den Web-Client zugreifen können, soll aber als Liste interner Funktionen betrachtet werden. Auf die Sicht des Benutzer wird in Abschnitt 3.6 genauer eingegangen.

Funktion: `list`

Parameter:

`start` Dokumenten-ID, Eingabe, optional;
`max` positive ganze Zahl, Eingabe, optional

Beschreibung: Diese Funktion liefert eine Liste der Dokumente mit einer Dokumenten-ID größer `start` bzw. 0, jedoch maximal `max` bzw. unbegrenzt viele Dokumente zurück.

Funktion: `get_structure`

Parameter:

`id` Dokumenten-ID, Eingabe

Beschreibung: Diese Funktion liefert die Strukturinformationen zum Dokument `id` zurück.

Funktion: `get_metadata`

Parameter:

`id` Dokumenten-ID, Eingabe;
`keys` Liste von Zeichenketten, Eingabe, optional

Beschreibung: Diese Funktion gibt die zum Dokument `id` gespeicherten Metadaten mit den Schlüsseln `keys` zurück.

Funktion: `get_history`

Parameter:

`id` Dokumenten-ID, Eingabe

Beschreibung: Diese Funktion ermittelt zum Dokument `id` die Geschichte, das heißt Infomationen über alle Versionen und die Benutzer, die das Dokument verwendet haben.

Funktion: `change_metadata`

Parameter:

`id` Dokumenten-ID, Eingabe;
`metadata` Liste mit Schlüssel-Wert-Paaren, Eingabe

Beschreibung: Diese Funktion setzt die in `metadata` spezifizierten Metadaten im Dokument `id`.

Funktion: `checkout_for_view`

Parameter:

`id` Dokumenten-ID, Eingabe;
`version` positive ganze Zahl, Eingabe, optional;
`paths` Liste von Pfaden, Eingabe, optional;
`output_module_spec` Spezifikation des Ausgabemoduls, Eingabe

Beschreibung: Diese Funktion kopiert aus dem Dokument `id` die in `paths` referenzierten bzw. alle Dateien der Version `version` in den durch `output_module_spec` beschriebenen Benutzerbereich. Ein Zurückspeichern ist nicht möglich.

Funktion: `checkout_for_update`

Parameter:

`id` Dokumenten-ID, Eingabe;
`version` positive ganze Zahl, Eingabe, optional;
`paths` Liste von Pfaden, Eingabe, optional;
`output_module_spec` Spezifikation des Ausgabemoduls, Eingabe;
`lock_token` Sperrschlüssel, Ausgabe;
`description` Zeichenkette, Eingabe, optional

Beschreibung: Diese Funktion kopiert aus dem Dokument `id` die in `paths` referenzierten bzw. alle Dateien der Version `version` in den durch `output_module_spec` beschriebenen Benutzerbereich. Zum Zurückspeichern ist `lock_token` und eine zugriffsberechtigte Benutzeridentität notwendig. Wenn das Dokument bereits von einem Benutzer zum Ändern gesperrt wurde, wird der zugehörige `lock_token` zurückgeliefert. Das Ziel der Bearbeitung kann in `description` beschrieben werden.

Funktion: `checkin`

Parameter:

`id` Dokumenten-ID, Eingabe;
`paths` Liste von Pfaden, Eingabe, optional;
`input_module_spec` Spezifikation des Eingabemoduls, Eingabe;
`lock_token` Sperrschlüssel, Eingabe;
`description` Zeichenkette, Eingabe;
`partial` boolscher Wert, Eingabe

Beschreibung: Diese Funktion speichert das Dokument `id` aus dem durch `input_module_spec` beschriebenen System zurück. Dabei muss der Sperrschlüssel `lock_token` und eine Beschreibung `description` angeben werden. Optional können auch nur bestimmte Teile (`partial` ist wahr, `paths`) zurückgespeichert werden. Der Sperrschlüssel bleibt für die noch nicht zurückgeschriebenen Teile weiter gültig.

Funktion: `cancel_checkout`

Parameter:

`id` Dokumenten-ID, Eingabe;
`paths` Liste von Pfaden, Eingabe, optional;
`lock_token` Sperrschlüssel, Eingabe;
`description` Zeichenkette, Eingabe

Beschreibung: Diese Funktion bricht das Bearbeiten des Dokuments `id` ab. Dabei muss der Sperrschlüssel `lock_token` und eine Beschreibung `description` angeben werden. Optional können auch nur bestimmte Teile (`paths`) der Bearbeitung abgebrochen werden. Der Sperrschlüssel bleibt für die nicht freigegebenen Teile weiter gültig.

Funktion: `create_document`

Parameter:

`id` Dokumenten-ID, Ausgabe;
`name` Zeichenkette, Eingabe;
`paths` Liste von Pfaden, Eingabe, optional;
`input_module_spec` Spezifikation des Eingabemoduls, Eingabe, optional;
`copy_of` Dokumenten-ID, Eingabe, optional

Beschreibung: Diese Funktion legt ein Dokument mit dem Namen `name` an und speichert als erste Version optional die Daten, die durch `paths` spezifiziert und über `input_module_spec` geladen werden. Als Rückgabe wird die Dokumenten-Identifikationsnummer `id` geliefert. Optional kann auch das Dokument `copy_of` dupliziert werden. Dann darf jedoch keine `input_module_spec` angegeben sein.

Funktion: `delete_document`

Parameter:

`id` Dokumenten-ID, Eingabe

Beschreibung: Diese Funktion löscht das Dokument vollständig. Dieser Vorgang ist nicht umkehrbar.

Funktion: `finalize_document`

Parameter:

`id` Dokumenten-ID, Eingabe;
`description` Zeichenkette, Eingabe;
`keep_until` Datum, Eingabe, optional;
`auto_delete` boolscher Wert, Eingabe, optional;
`delete_history` boolscher Wert, Eingabe, optional

Beschreibung: Diese Funktion schließt das Dokument `id` ab. Nach dem Abschluss kann das Dokument nicht mehr bearbeitet werden. Das Dokument muss bis zum Zeitpunkt `keep_until` aufbewahrt werden. Optional kann es beim Erreichen des Zeitpunkts gelöscht werden (`auto_delete`). Ist `delete_history` wahr, so wird die Geschichte des Dokuments, das heißt das Protokoll und alle Versionen vor der aktuellen Version, gelöscht.

Funktion: `get_input_module_list`

Parameter: keine

Beschreibung: Diese Funktion liefert eine Liste aller verfügbaren Eingabemodule und Informationen zu deren Parametern.

Funktion: `get_output_module_list`

Parameter: keine

Beschreibung: Diese Funktion liefert eine Liste aller verfügbaren Ausgabemodule und Informationen zu deren Parametern.

Funktion: `get_rights_list`

Parameter: keine

Beschreibung: Diese Funktion liefert eine Liste aller vom Modul verwendeten Zugriffsrechte-Eigenschaften.

A.2 Modul zur Benutzerverwaltung

Folgende Funktionen werden zusätzlich zur Authentifizierung vom Modul zur Verfügung gestellt:

Funktion: `list_users`

Parameter:

`start` Benutzer-Identifikation, Eingabe, optional;
`max` Ganzzahl größer 0, Eingabe, optional

Beschreibung: Diese Funktion listet alle Benutzer des Systems auf. Wenn angegeben, werden maximal `max` Benutzer ab dem Benutzer `start` gelistet.

Funktion: `create_user`

Parameter:

`id` Benutzer-Identifikation, Ausgabe;
`name` Benutzer-Kennung als Zeichenkette, Eingabe

Beschreibung: Diese Funktion legt einen neuen Benutzer mit dem Kürzel `name` an und gibt dessen Benutzer-ID zurück. Der Benutzer hat noch keinerlei Berechtigungen und ist auf nicht aktiv geschalten. Werden Benutzerdaten ausschließlich in Fremdsystemen gespeichert, ist dies unter Umständen nicht möglich.

Funktion: `delete_user`

Parameter:

`id` Benutzer-Identifikation, Ausgabe

Beschreibung: Diese Funktion löscht den Benutzer `id` aus dem System. Werden Benutzerdaten ausschließlich in Fremdsystemen gespeichert, ist dies nicht möglich.

Funktion: `change_user`

Parameter:

`id` Benutzer-Identifikation, Eingabe;
`name` Benutzer-Kennung als Zeichenkette, Eingabe

Beschreibung: Diese Funktion ändert die Benutzerkennung des Benutzers `id` auf `name`. Die alte Kennung geht verloren.

Funktion: `user_info`

Parameter:

`id` Benutzer-Identifikation, Eingabe

Beschreibung: Diese Funktion ermittelt alle benutzerbezogen gespeicherten Daten zum Benutzer und liefert die zurück.

Funktion: `list_groups`

Parameter:

`start` Gruppen-Identifikation, Eingabe, optional;
`max` Ganzzahl größer 0, Eingabe, optional

Beschreibung: Diese Funktion listet alle Gruppen des Systems auf. Wenn angegeben, werden maximal `max` Gruppen ab der Gruppe `start` gelistet.

Funktion: `create_group`

Parameter:

`id` Gruppen-Identifikation, Ausgabe;
`name` Gruppen-Kennung als Zeichenkette, Eingabe

Beschreibung: Diese Funktion legt eine neue Gruppe mit dem Kürzel `name` an und gibt deren Gruppen-ID zurück. Die Gruppe hat noch keinerlei Berechtigungen und ist auf nicht aktiv geschalten. Gruppen werden nicht in Fremdsystemen gespeichert.

Funktion: `delete_group`

Parameter:

`id` Gruppen-Identifikation, Ausgabe

Beschreibung: Diese Funktion löscht die Gruppe `id` aus dem System.

Funktion: `change_group`

Parameter:

`id` Gruppen-Identifikation, Eingabe;
`name` Gruppen-Kennung als Zeichenkette, Eingabe

Beschreibung: Diese Funktion ändert die Gruppenkennung der Gruppe `id` auf `name`. Die alte Kennung geht verloren.

Funktion: `group_info`

Parameter:

`id` Gruppen-Identifikation, Eingabe

Beschreibung: Diese Funktion ermittelt alle gruppenbezogen gespeicherten Daten zur Gruppe und liefert sie zurück.

Funktion: `get_rights`

Parameter:

`id` Gruppen- oder Benutzer-Identifikation, Eingabe;
`rights_list` Liste von Zeichenketten, optional, Eingabe

Beschreibung: Diese Funktion liefert die Berechtigungen zu dem Benutzer oder der Gruppe `id`. Falls spezifiziert, werden nur die in `rights_list` spezifizierten Rechte ermittelt. Die Funktion liefert nur allgemeine Rechte des Benutzers oder der Gruppe, aber keine dokumentspezifischen. Die Rückgabe erfolgt als Zeichenketten bzw. Binärblöcke.

Funktion: `set_right`

Parameter:

`id` Gruppen- oder Benutzer-Identifikation, Eingabe;
`right` Zeichenkette, Eingabe;
`value` Binärdatenblock oder Zeichenkette, Eingabe

Beschreibung: Diese Funktion setzt für den Benutzer oder die Gruppe `id` das Recht `right` mit dem Wert `value`. Diese Daten werden benutzer- bzw. gruppenbezogen und nicht dokumentbezogen gespeichert.

Funktion: `get_rights_list`

Parameter: keine

Beschreibung: Diese Funktion ermittelt die Rechte, die vom Benutzermodul verwaltet werden und liefert sie als Liste inklusive einer Beschreibung zurück. Dabei handelt es sich nur um die Rechte, die im Zusammenhang mit Benutzern und Gruppen, jedoch nicht mit Dokumenten relevant sind.

Funktion: `get_rights_document`

Parameter:

`id` Gruppen- oder Benutzer-Identifikation, Eingabe;
`did` Dokumenten-ID, Eingabe;
`rights_list` Liste von Zeichenketten, optional, Eingabe

Beschreibung: Diese Funktion liefert die Berechtigungen zu dem Benutzer oder der Gruppe `id` für das Dokument `did`. Falls spezifiziert, werden nur die in `rights_list` spezifizierten Rechte ermittelt. Die Funktion liefert nur allgemeine Rechte des Benutzers oder der Gruppe, aber keine dokumentspezifischen. Die Rückgabe erfolgt als Zeichenketten bzw. Binärblöcke.

Funktion: `set_right_document`

Parameter:

`id` Gruppen- oder Benutzer-Identifikation, Eingabe;
`did` Dokumenten-ID, Eingabe;
`right` Zeichenkette, Eingabe;
`value` Binärdatenblock oder Zeichenkette, Eingabe

Beschreibung: Diese Funktion setzt für den Benutzer oder die Gruppe `id` und das Dokument `did` das Recht `right` mit dem Wert `value`. Diese Daten werden benutzer- bzw. gruppenbezogen und nicht dokumentbezogen gespeichert.

Funktion: `get_rights_list_document`

Parameter: keine

Beschreibung: Diese Funktion ermittelt die Rechte, die vom Benutzermodul im Zusammenhang mit Dokumenten und Benutzern/Gruppen verwaltet werden und liefert sie als Liste inklusive einer Beschreibung zurück.

Funktion: `get_auth_methods_list`

Parameter: keine

Beschreibung: Diese Funktion liefert eine Liste aller verfügbaren Mechanismen zu Authentifizierung inklusive deren Parameter.

Funktion: `auth`

Parameter:

`method` Zeichenkette, Eingabe;
`parameters` Liste von Parametern, Eingabe

Beschreibung: Diese Funktion führt eine Authentifizierung mit der angegebenen Methode `method` und den zugehörigen Paramatern `parameters` durch.

A.3 Modul zur Datenspeicherung

Die Funktionen der Datenspeicherung sind wie folgt definiert:

Funktion: `list_blocks`

Parameter:

`modulid` Zeichenkette, Eingabe;
`start` Blocknummer, Eingabe, optional;
`max` positiv ganze Zahl, Eingabe, optional

Beschreibung: Listet alle bzw. maximal `max` Blöcke ab `start` für das Modul `modulid` auf.

Funktion: `get_block`

Parameter:

`modulid` Zeichenkette, Eingabe;
`block` Blocknummer, Eingabe

Beschreibung: Ermittelt den Datenblock `block` des Moduls `modulid` und liefert ihn zurück.

Funktion: `set_block`

Parameter:

`modulid` Zeichenkette, Eingabe;
`block` Blocknummer, Eingabe;
`data` Binärdatenblock, Eingabe

Beschreibung: Setzt den Block `block` des Moduls `modulid` auf `data`.

Funktion: `delete_block`

Parameter:

`modulid` Zeichenkette, Eingabe;
`block` Blocknummer, Eingabe

Beschreibung: Löscht den Block `block` des Moduls `modulid`.

Funktion: `create_block`

Parameter:

`modulid` Zeichenkette, Eingabe;
`block` Blocknummer, Ausgabe;
`data` Binärdatenblock, Eingabe

Beschreibung: Erstellt einen Block für das Modul `modulid` und setzt ihn auf `data`. Die Blocknummer wird in `block` geschrieben.

A.4 Modul für Groupware-Funktionalität

A.4.1 Erweitertungen von Funktionen der Kernmodule

Im folgenden sind Funktionen der Kernmodule beschrieben, die modifiziert werden. Dabei ist die Auflistung nach den hinzugefügten Funktionalitäten zusammengefasst.

Funktionen: `change_metadata`, `checkout_for_update`, `checkin`, `cancel_checkout`, `create_document`, `delete_document`, `finalize_document`

zusätzliche Parameter:

`notify_users` Liste von Benutzer-IDs, Eingabe, optional;
`notify_message` Zeichenkette, Eingabe, optional

Beschreibung: Optional können mit dieser Erweiterung die Benutzer `notify_users` über die Durchführung der Aktion mit `notify_message` benachrichtigt werden.

Funktionen: `change_metadata`, `checkout_for_update`, `checkin`, `cancel_checkout`, `create_document`, `delete_document`, `finalize_document`

zusätzliche Parameter:

`on_success_action` XML-Befehlsblock, Eingabe, optional;
`on_failure_action` XML-Befehlsblock, Eingabe, optional

Beschreibung: Die angegebenen Befehlsblöcke werden bei erfolgreicher (`on_success_action`) bzw. bei erfolgloser oder fehlerhafter Abarbeitung (`on_failure_action`) durch den Dispatcher ausgeführt.

Funktion: `checkout_for_update`

zusätzliche Parameter:

`deadline` Datum, Eingabe, Eingabe, optional;
`auto_checkin` boolscher Wert, Eingabe, optional;
`paths` Liste von Pfaden, Eingabe, optional;
`input_module_spec` Spezifikation des Eingabemoduls, Eingabe;
`lock_token` Sperrschlüssel, Eingabe;
`description` Zeichenkette, Eingabe, optional;
`warn` Zeitintervall, Eingabe, optional;
`auto_checkin` Befehl zum Rückspeichern, Eingabe, optional;
`auto_cancel` Befehl zum Abbrechen, Eingabe, optional;
`deadline_notify_users` Liste von Benutzer-IDs, Eingabe, optional;
`deadline_notify_message` Zeichenkette, Eingabe, optional;
`deadline_action` XML-Befehlsblock, Eingabe, optional

Beschreibung: Der Parameter `deadline` gibt den Zeipunkt an, an dem die Bearbeitung abgeschlossen sein muss. Der bearbeitende Benutzer wird auf den Ablauf der Frist zum Zeitpunkt `deadline - warn` durch eine Nachricht darauf hingewiesen. Ist `auto_checkin` angegeben, so wird das Dokument – sofern möglich – automatisch zurückgespeichert. Entsprechend wird mit `auto_cancel` der Abbruch der Bearbeitung veranlasst. Die Parameter dazu sind analog zu denen im Befehl. Die in `deadline_notify_users` angegebenen Benutzer werden mit der Nachricht `deadline_notify_message` auf den Ablauf der Bearbeitungsfrist hingewiesen. Ist `deadline_action` angegeben, so wird dieser Block bei Ablauf der Frist vom Dispatcher ausgeführt.

A.4.2 Funktionen des Moduls

Das Groupware-Modul fügt die im Folgenden beschriebenen Befehle hinzu.

Funktion: `send_message`

zusätzliche Parameter:

`ids` Liste mit Benutzer-ID, Eingabe;
`text` Zeichenkette, Eingabe;
`document_list` Liste mit Dokumenten-IDs, Eingabe, optional;
`on_receive_action` XML-Befehlsblock, Eingabe, optional

Beschreibung: Diese Funktion sendet eine Nachricht an die Benutzer `ids` mit dem Inhalt `text`. Die Nachricht beinhaltet die Dokumente aus `document_list`. Beim Empfang wird für den jeweiligen Benutzer der Befehl `on_receive_action` ausgeführt.

Funktion: `get_message_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`start` positive ganze Zahl, Eingabe, optional;
`max` positive ganze Zahl, Eingabe, optional;
`only_new` boolscher Wert, Eingabe, optional

Beschreibung: Diese Funkion liefert die Ereignis- bzw. Nachrichtenliste für den Benutzer `id` zurück. Es werden maximal `max` Einträge ab Position `start` zurückgegeben. Ist `only_new` angegeben, werden nur neue Nachrichten angezeigt.

Funktion: `get_message`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`number` positive ganze Zahl, Eingabe;
`delete` boolscher Wert, Eingabe, optional

Beschreibung: Dieser Eintrag liefert die Nachricht `number` des Benutzers `id` zurück. Falls `delete` wahr ist, wird die Nachricht aus der Liste entfernt.

Funktion: `get_comments`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`start` positive ganze Zahl, Eingabe, optional;
`max` positive ganze Zahl, Eingabe, optional

Beschreibung: Diese Funktion lädt die ersten `max` Themen ab Position `start` für das Dokument `id`.

Funktion: `add_comment`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`parent` positive ganze Zahl, Eingabe, optional;
`message` Zeichenkette, Eingabe

Beschreibung: Diese Funktion fügt `message` als Nachricht im Diskussionsforum des Dokuments `id` hinzu. Optional kann eine Nachrichtennummer `parent` angegeben werden, auf die sich die Nachricht bezieht.

Funktion: `change_deadline`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`deadline` Datum, Eingabe, Eingabe, optional;
`auto_checkin` Befehl zum Rückspeichern, Eingabe, optional;
`auto_cancel` Befehl zum Abbrechen, Eingabe, optional;
`warn` Zeitintervall, Eingabe, optional;
`deadline_notify_users` Liste von Benutzer-IDs, Eingabe, optional;
`deadline_notify_message` Zeichenkette, Eingabe, optional;
`deadline_action` XML-Befehlsblock, Eingabe, optional

Beschreibung: Dieser Befehl ändert eine Bearbeitungsfrist. Die Bedeutung der Parameter ist identisch mit denen in `checkout_for_update`. Die alte Frist wird gelöscht.

Funktion: `request_document`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`user_id` Benutzer-ID, Eingabe;
`description` Zeichenkette, Eingabe;
`auto_checkout_read` boolscher Wert, Eingabe, optional;
`auto_checkout_update` boolscher Wert, Eingabe, optional;
`version` positive ganze Zahl, Eingabe, optional;
`paths` Liste von Pfaden, Eingabe, optional;
`output_module_spec` Spezifikation des Ausgabemoduls, Eingabe;
`lock_token` Sperrschlüssel, Ausgabe;
`description` Zeichenkette, Eingabe, optional

Beschreibung: Diese Funktion meldet dem aktuellen Bearbeiter, dass der Benutzer `user_id` das Dokument `id` bearbeiten möchte. Die Gründe sind in `description` beschrieben. Sollte das Dokument verfügbar werden, dann wird es optional zum Lesen (`auto_checkout_read`) oder Schreiben (`auto_checkout_update`) geladen. Die Parameter dazu sind dieselben wie beim Befehl `checkout_for_read` bzw. `checkout_for_update`.

Funktion: `watch_document`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`users` Liste von Benutzer-IDs, Eingabe;
`operations` Liste mit Zeichenketten, Eingabe;
`start` Datum, Eingabe, optional;
`end` Datum, Eingabe, optional;
`users` Liste mit Benutzer-IDs, Eingabe, optional;

Beschreibung: Diese Funktion setzt die Überwachungsfunktion des Benutzers `users` für das Dokument `id` neu. Es werden die Funktionen `operations` zwischen den Zeitpunkten `start` und `end` der Benutzer `users` überwacht. Der Benutzer erhält eine Nachricht, falls eine dieser Funktionen ausgelöst wurde.

Funktion: `unwatch_document`

zusätzliche Parameter:

`id` Dokumenten-ID, Eingabe;
`users` Liste von Benutzer-IDs, Eingabe

Beschreibung: Diese Funktion löscht vorhandene Überwachungsfunktionen der Benutzer `users` für das Dokument `id`.

Funktion: `get_personal_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`name_list` Zeichenketten, Eingabe, optional

Beschreibung: Diese Funktion lädt die für den Benutzer `id` vorhandene Dokumentliste mit dem Namen `name_list`. Ist keine Benutzer-ID angegeben, dann wird die Funktion für den ausführenden Benutzer durchgeführt.

Funktion: `create_personal_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`name` Zeichenkette, Eingabe, optional

Beschreibung: Diese Funktion legt für den Benutzer `id` bzw. den ausführenden Benutzer eine neue Dokumentliste `name` an.

Funktion: `delete_personal_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`name_list` Liste von Zeichenketten, Eingabe

Beschreibung: Diese Funktion löscht die in `name_list` angegebenen Listen des Benutzers `id`.

Funktion: `add_to_personal_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`name` Zeichenkette, Eingabe;
`path` Zeichenkette, Eingabe;
`directory` boolscher Wert, Eingabe;
`document` Dokumenten-ID, Eingabe;
`description` Zeichenkette, Eingabe, optional

Beschreibung: Diese Funktion fügt `path` in der Dokumentenliste `name` des Benutzers `id` einen neuen Eintrag ein. Ist `directory` wahr, so wird ein Verzeichnis erstellt, ansonsten wird das Dokument `document` mit der Beschreibung `description` eingefügt.

Funktion: `remove_from_personal_list`

zusätzliche Parameter:

`id` Benutzer-ID, Eingabe, optional;
`name` Zeichenkette, Eingabe;
`path` Zeichenkette, Eingabe

Beschreibung: Diese Funktion löscht den Eintrag `path` aus der Liste `name` des Benutzers `id`.

Bei einigen der oben definierten Befehle kommen noch Funktionen hinzu:

Funktionen: `request_document`, `watch_document`, `send_message`

zusätzliche Parameter:

`on_success_action` XML-Befehlsblock, Eingabe, optional;
`on_failure_action` XML-Befehlsblock, Eingabe, optional

Beschreibung: Die angegebenen Befehlsblöcke werden bei erfolgreicher (`on_success_action`) bzw. bei erfolgloser oder fehlerhafter Abarbeitung (`on_failure_action`) durch den Dispatcher ausgeführt.

Funktionen: `request_document`, `watch_document`, `add_to_personal_list`, `remove_from_personal_list`, `create_personal_list`, `delete_personal_list`, `add_comment`

zusätzliche Parameter:

`notify_users` Liste von Benutzer-IDs, Eingabe, optional;
`notify_message` Zeichenkette, Eingabe, optional

Beschreibung: Optional können mit dieser Erweiterung die Benutzer `notify_users` über die Ausführung der Aktionen mit `notify_message` benachrichtigt werden.

B XML-Formate

B.1 Schemata der XML-Kommunikation

Die in diesem Anhang aufgeführten XML-Schemata definieren analog zu den in Anhang A aufgelisteten Funktionen die entsprechenden XML-Anfragen der einzelnen Module. Des weiteren wird zunächst in Teil B.1.1 dieses Anhangs die XML-Anfragesprache definiert, in die die danach definierten Anfragen eingebettet werden.

Sämtliche aufgeführten Dateien befinden sich auf der begelegten CD im Verzeichnis `schemata` und sind aus Gründen der Übersichtlichkeit nicht abgedruckt.

B.1.1 Wrapper

Anfragen: `request.xsd`
Antworten: `response.xsd`

B.1.2 Kernfunktionen

XML-Datentypen: `core-datatypes.xsd`
Anfragen: `core-request.xsd`
Antworten: `core-response.xsd`

B.1.3 Authentifizierung

Anfragen: `auth-request.xsd`
Antworten: `auth-response.xsd`

B.1.4 Benutzerverwaltung

XML-Datentypen: `user-datatypes.xsd`
Anfragen: `user-request.xsd`
Antworten: `user-response.xsd`

B.1.5 Ein- und Ausgabemodule

Eingabemodule: `input-module-spec.xsd`
Ausgabemodule: `output-module-spec.xsd`

B.1.6 Groupware-Modul

XML-Datentypen: `groupware-datatypes.xsd`
Anfragen: `groupware-request.xsd`
Antworten: `groupware-response.xsd`

C Quelltexte

Dieser Anhang enthält ausgewählte Quelltexte des Dokumentenmanagementsystems. Sämtliche aufgeführten Dateien befinden sich ebenfalls auf der begelegten CD im Verzeichnis `java-src` in der Java-üblichen Package-Struktur.

C.1 Zugriff auf Dokumente

`DocumentStorage.java`

```
1  package com.ewerk.dms.storage;
2
3  public interface DocumentStorage {
4
5      /** Diese Methode baut eine Verbindung zum Datenspeicher auf. Sie wird
6          stets zuerst aufgerufen. */
7      public void connect();
8
9      /** Diese Methode wird abschließend zu allen Aktionen aufgerufen */
10     public void disconnect();
11
12     /** Diese Methode gibt an, ob Transaktionen untertützt werden.
13         @return true, falls ja, sonst false
14     */
15     public boolean supportsTransactions();
16
17     /** Diese Methode startet eine neue Transaktion. Sind noch Transaktionen
18         offen, die nicht mit commit beendet wurden, so werden diese mit
19         einem Rollback beendet.
20     */
21     public void beginTransaction();
22
23     /** Diese Methode beendet eine Transaktion.
24         @param commit true, falls Änderungen gespeichert werden sollen.
25     */
26     public void endTransaction(boolean commit);
27
28     /** Diese Methode erstellt ein neues Dokument
29         @return Dokumenten-ID
30     */
```

```
31      public int createDocument();
32
33      /** Diese Methode lädt eine Blockliste des Dokuments.
34          @param document Dokumenten-ID
35          @return Liste der Blocknummern als Feld
36      */
37      public int[] getBlocksOfDocument(int document);
38
39      /** Diese Methode entfernt ein Dokument vollständig.
40          @param document Dokumenten-ID
41      */
42      public void removeDocument(int document);
43
44      /** Diese Methode fügt einen Block zu einem Dokument hinzu.
45          @param document Dokumenten-ID
46          @param data Datenblock
47          @return Blocknummer
48      */
49      public int addBlockToDocument(int document, byte[] data);
50
51      /** Diese Methode löscht einen Block aus einem Dokument.
52          @param document Dokumenten-ID
53          @param block Blocknummer
54      */
55      public void removeBlockFromDocument(int document, int block);
56
57      /** Diese Methode ändert einen Block in einem Dokument.
58          @param document Dokumenten-ID
59          @param data Datenblock
60          @param block Blocknummer
61      */
62      public void updateBlockOfDocument(int document, int block, byte[] data);
63
64      /** Diese Methode lädt einen Block.
65          @param document Dokumenten-ID
66          @param block Blocknummer
67          @return Datenblock als Feld
68      */
69      public byte[] loadBlockOfDocument(int document, int block);
70  }
```

C.2 Zugriff auf Konfigurationen

ConfigurationStorage.java

```
1  package com.ewerk.dms.storage;
2
3  /** Diese Schnittstelle stellt den Zugriff auf modulspezifische
4      Konfigurationen bereit. Jedes Modul kann unter seinem Modulnamen
5      beliebige Daten speichern.
6  */
7  public interface ConfigurationStorage {
8
9      /** Wird vor der Arbeit mit einer Instanz aufgerufen. */
10     public void init();
11
12     /** Wird zum Abschluss der Arbeit mit einer Instanz aufgerufen. */
13     public void finish();
14
15     /** Diese Methode erstellt eine neue Konfiguration.
16         @param module Name des Moduls
17         @return ID der Konfiguration
18     */
19     public int createConfiguration(String module);
20
21     /** Diese Methode speichert eine Konfiguration.
22         @param module Name des Moduls
23         @param configuration ID der Konfiguration
24         @param data Datenblock
25     */
26     public void storeConfiguration(String module, int configuration,
27                 byte[] data);
28
29     /** Diese Methode lädt eine Konfiguration.
30         @param module Name des Moduls
31         @param configuration ID der Konfiguration
32         @return Datenblock
33     */
34     public byte[] loadConfiguration(String module, int configuration);
35
36     /** Diese Methode löscht eine Konfiguration.
37         @param module Name des Moduls
38         @param configuration ID der Konfiguration
39     */
40     public void deleteConfiguration(String module, int configuration);
41 }
```

C.3 Modulschnittstelle für Benutzerspeicher

UserStorage.java

```
1  package com.ewerk.dms.user;
2
3  /** Das Interface UserStorage dient zur Anbindung des DMS an
4      Verzeichnisdienste oder Datenbanken, um die Benutzerdaten
5      zu speichern.
6  */
7  public interface UserStorage {
8
9      /** Diese Methode baut eine Verbindung zum Datenspeicher auf. Sie wird
10         stets zuerst aufgerufen. */
11     public void connect();
12
13     /** Diese Methode wird abschließend zu allen Aktionen aufgerufen */
14     public void disconnect();
15
16     /** Diese Methode gibt an, ob Transaktionen untertützt werden.
17         @return true, falls ja, sonst false
18     */
19     public boolean supportsTransactions();
20
21     /** Diese Methode startet eine neue Transaktion. Sind noch Transaktionen
22         offen, die nicht mit commit beendet wurden, so werden diese mit
23         einem Rollback beendet.
24     */
25     public void beginTransaction();
26
27     /** Diese Methode beendet eine Transaktion.
28         @param commit true, falls Änderungen gespeichert werden sollen.
29     */
30     public void endTransaction(boolean commit);
31
32     /** Diese Methode erstellt einen neuen Benutzer.
33         @param name Kürzel oder Name des Benutzers.
34         @return ID des Benutzers
35     */
36     public int createUser(String name);
37
38     /** Diese Methode löscht einen Benutzer aus dem System. Bei einer
39         Anbindung an vorhandene Verzeichnisdienste sollte beachtet werden,
40         dass der Benutzer nur aus dem DMS-spezifischen Teil gelöscht wird
41         und andere Daten nicht verändert werden.
42         @param user ID des Benutzers
43     */
44     public void deleteUser(int user);
45
46     /** Diese Methode lädt eine Eigenschaft eines Benutzers.
47         @param user ID des Benutzers
```

```
48          @param name Name der Eigenschaft
49          @return Wert der Eigenschaft
50      */
51      public String getUserProperty(int user, String name);
52
53      /** Diese Methode lädt eine Liste aller Eigenschaften eines Benutzers.
54          @param user ID des Benutzers
55          @return Liste der Eigenschaften
56      */
57      public String[] getUserProperties(int user);
58
59      /** Diese Methode speichert eine Eigenschaft eines Benutzers.
60          @param user ID des Benutzers
61          @param name Name der Eigenschaft
62          @param value Wert der Eigenschaft
63      */
64      public void putUserProperty(int user, String name, String value);
65
66      /** Diese Methode löscht eine Eigenschaft eines Benutzers.
67          @param user ID des Benutzers
68          @param name Name der Eigenschaft
69      */
70      public void removeUserProperty(int user, String name);
71  }
```

C.4 Zugriff auf Benutzerdaten

UserManager.java

```
1  package com.ewerk.dms.user;
2
3  /**
4      Die Schnittstelle UserManager dient zum Zugriff auf die Benutzerdaten.
5      Sie ist vereinfacht den Zugriff gegenüber UserStorage erheblich und
6      darf von allen Module des Systems verwendet werden.
7  */
8  public interface UserManager {
9
10     /** Diese Methode erstellt einen neuen Benutzer.
11         @param name Kürzel oder Name des Benutzers.
12         @return ID des Benutzers
13     */
14     public int createUser(String name);
15
16     /** Diese Methode löscht einen Benutzer aus dem System.
17         @param user ID des Benutzers
18     */
19     public void deleteUser(int user);
20
21     /** Diese Methode liefert die ID des ausführenden Benutzers zurück.
22         @return ID des Benutzers
23     */
24     public int getUser();
25
26     /** Diese Methode ermittelt die Gruppen, denen der Benutzer angehört.
27         @return Liste der Gruppen
28     */
29     public int[] getGroups();
30
31     /** Diese Methode ermittelt die Gruppen, denen der Benutzer angehört.
32         @param user ID des Benutzers
33         @return Liste der Gruppen
34     */
35     public int[] getGroups(int user);
36
37     /** Diese Methode lädt eine Eigenschaft des Benutzers.
38         @param name Name der Eigenschaft
39         @return Wert der Eigenschaft
40     */
41     public String getProperty(String name);
42
43     /** Diese Methode lädt eine Eigenschaft eines Benutzers.
44         @param user ID des Benutzers
45         @param name Name der Eigenschaft
46         @return Wert der Eigenschaft
47     */
```

```
48      public String getProperty(int user, String name);
49
50      /** Diese Methode lädt eine Liste aller Eigenschaften des Benutzers.
51          @return Liste der Eigenschaften
52      */
53      public String[] getProperties();
54
55      /** Diese Methode lädt eine Liste aller Eigenschaften eines Benutzers.
56          @param user ID des Benutzers
57          @return Liste der Eigenschaften
58      */
59      public String[] getProperties(int user);
60
61      /** Diese Methode speichert eine Eigenschaft des Benutzers.
62          @param name Name der Eigenschaft
63          @param value Wert der Eigenschaft
64      */
65      public void putProperty(String name, String value);
66
67      /** Diese Methode speichert eine Eigenschaft eines Benutzers.
68          @param user ID des Benutzers
69          @param name Name der Eigenschaft
70          @param value Wert der Eigenschaft
71      */
72      public void putProperty(int user, String name, String value);
73
74      /** Diese Methode fügt den Benutzer zu einer Gruppe hinzu.
75          @param group ID der Gruppe (ID des Benutzers)
76      */
77      public void addToGroup(int group);
78
79      /** Diese Methode fügt einen Benutzer zu einer Gruppe hinzu.
80          @param user ID des Benutzers
81          @param group ID der Gruppe (ID des Benutzers)
82      */
83      public void addToGroup(int user, int group);
84
85      /** Diese Methode löscht den Benutzer aus einer Gruppe.
86          @param group ID der Gruppe (ID des Benutzers)
87      */
88      public void removeFromGroup(int group);
89
90      /** Diese Methode löscht einen Benutzer aus einer Gruppe.
91          @param user ID des Benutzers
92          @param group ID der Gruppe (ID des Benutzers)
93      */
94      public void removeFromGroup(int user, int group);
95
96      /** Diese Methode ermittelt den Namen des Benutzers.
97          @return Name des Benutzers
98      */
```

```
 99      public String getName();
100
101      /** Diese Methode ermittelt den Namen eines Benutzers.
102          @param user ID des Benutzers
103          @return Name des Benutzers
104      */
105      public String getName(int user);
106
107      /** Diese Methode setzt ein Recht des Benutzers.
108          @param right Name des Rechts
109      */
110      public void setRight(String right);
111
112      /** Diese Methode setzt ein Recht eines Benutzers.
113          @param user ID des Benutzers
114          @param right Name des Rechts
115      */
116      public void setRight(int user, String right);
117
118      /** Diese Methode löscht ein Recht des Benutzers.
119          @param right Name des Rechts
120      */
121      public void removeRight(String right);
122
123      /** Diese Methode löscht ein Recht eines Benutzers.
124          @param user ID des Benutzers
125          @param right Name des Rechts
126      */
127      public void removeRight(int user, String right);
128
129      /** Diese Methode prüft ein Recht des Benutzers.
130          @param right Name des Rechts
131          @return true, falls er es besitzt
132      */
133      public boolean hasRight(String right);
134
135      /** Diese Methode prüft ein Recht eines Benutzers.
136          @param user ID des Benutzers
137          @param right Name des Rechts
138          @return true, falls er es besitzt
139      */
140      public boolean hasRight(int user, String right);
141
142  }
```

C.5 Modulschnittstelle

DispatcherClient.java

```
1  package com.ewerk.dms.module;
2
3  /** Das Interface DispatcherClient muss von Modulen implementiert werden,
4      damit der Dispatcher das Modul aufruft, wenn ein registrierter Namespace
5      vorgefunden wird.
6  */
7  public interface DispatcherClient {
8      /** Die Funktion process wird vom Dispatcher aufgerufen, wenn ein Request
9          mit einem auf das Modul registrierten Namespace vorgefunden wird.
10         Der Knoten node wird nach der Verarbeitung vom Dispatcher in der
11         Form weiterverarbeitet, d.h. an andere registrierte Module
12         weitergegeben. Die Funktion darf den DOM-Teilbaum somit modifizieren.
13
14         @param namespaceURI     Namespace für den der Aufruf erfolgte.
15         @param priority         Prioritätsstufe des Aufrufs.
16         @param node             der DOM-Teilbaum des Dokuments, der den Request
17                                 beschreibt, also das Element request aus der
18                                 Anfrage, das als Kindelement das Befehlselement
19                                 enthält.
20         @param allNodes         Die Liste der bereits erstellten Antworten und
21                                 der nach dem aktuellen Befehl noch zu
22                                 verarbeitenden Befehle in der entsprechenden
23                                 DOM-Struktur nach dem XML-Schema request.xsd.
24                                 allNodes darf manipuliert werden, um neue
25                                 Befehle durch den Dispatcher ausführen zu
26                                 lassen.
27
28         @return Rückgabewert ist das Ergebnis des Requests als Element
29                 <code>response</code>.
30     */
31     public org.w3c.dom.Node process(String namespaceURI, int priority,
32                     org.w3c.dom.Node node, org.w3c.dom.NodeList allNodes);
33
34     /** Aufruf beim Initialisieren des Dispatchers bzw. nach Anmeldung
35         des Moduls. Aufrufeanzahl von init und finish ist gleich. */
36     public void init();
37
38     /** Aufruf vor jedem Dispatcher-Durchgang. */
39     public void startDispatch();
40
41     /** Aufruf nach jedem Dispatcher-Durchgang. */
42     public void endDispatch();
43
44     /** Aufruf beim Beenden des Dispatchers. Aufrufeanzahl von init und
45         finish ist gleich. */
46     public void finish();
47 }
```

C.6 Modulschnittstelle zur Authentifizierung

AuthenticationModule.java

```
1  package com.ewerk.dms.module;
2
3  import org.w3c.dom.Node;
4
5  /**
6      Das Interface AuthenticationModule muss von Modulen implementiert werden,
7      damit der Dispatcher das Modul zur Authentifizierung aufruft. Das Modul
8      muss ebenfalls das Interface DispatcherClient implementieren.
9  */
10 public interface AuthenticationModule {
11     /** Diese Methode wird vom Dispatcher zur Durchführung der
12         Authentifizierung aufgerufen. Diese Methode wird auch
13         aufgerufen, wenn das Modul diese Art der Authentifizierung
14         nicht implementiert.
15         @param node Der DOM-Baum der Parameter der Authentifizierung.
16         @return true bei erfolgreicher Authentifizierung, sonst false
17     */
18     public boolean authenticate(org.w3c.dom.Node node);
19 }
```

C.7 Dispatcher

Dispatcher.java

```
1  package com.ewerk.dms.dispatcher;
2
3  import java.util.*;
4  import com.ewerk.dms.module.*;
5  import com.ewerk.dms.storage.*;
6  import com.ewerk.dms.tss.*;
7
8  /**
9      Die Klasse Dispatcher ist die zentrale Einheit des DMS. Sie leitet
10     die Anfragen an die Module weiter und fasst deren Antworten zusammen.
11     Sie dient ebenfalls dazu, die Klassen zum Zugriff auf Dokumentdaten,
12     Task Schduling Service, Benutzerdaten und -rechte, etc zu erhalten.
13 */
14 public class Dispatcher {
15
16     /* ... */
17
18     /**
19         Liefert eine Instanz des Dispatchers zurück.
20         @return Dispatcher-Instanz
21     */
22     public static Dispatcher getInstance() { }
23
24     /**
25         Diese Funktion registriert ein Modul auf einen Namespace. Die
26         angegebene Priorität gibt an, in welcher Reihenfolge die Module
27         aufgerufen werden. Der Wertebereich umfasst den des Datentyps
28         <code>int</int>. Module können mit mehreren Prioritäten registriert
29         werden und werden dann mehrfach aufgerufen.
30         @param client       DispatcherClient des Moduls
31         @param namespaceURI Namespace, auf den registriert wird
32         @param priority     Priorität des Aufrufs
33     */
34     public void registerModule(DispatcherClient client, String namespaceURI,
35                     int priority) {}
36
37     /**
38         Einfache Version von <code>registerModule</code>. Priorität ist 0.
39         @see registerModule
40     */
41     public void registerModule(DispatcherClient client, String namespaceURI) {}
42
43
44     public void registerAuthenticationModule(
45                     AuthenticationModule authenticationModule) {}
46
47     /**
```

```
48          Dispatcht den das übergebene Dokument und gibt das Ergebnis als
49          Document zurück. Das Dokument sollte gegen das Schema
50          <code>request.xsd</code> geprüft sein. Das Ergebnis entspricht
51          dem Schema <code>response.xsd</code>
52
53          @return Ergebnis-Dokument.
54      */
55      public org.w3c.dom.Document dispatch(org.w3c.dom.Document document) {}
56
57      public TaskSchedulingService getTaskSchedulingService() {}
58
59      public DocumentStorage getDocumentStorage() {}
60
61  }
```

C.8 Task-Scheduling-Service

Dispatcher.java

```
1  package com.ewerk.dms.module;
2
3  import java.util.*;
4
5  public interface TaskSchedulingService {
6
7      /**
8          Diese Methode registriert die Abarbeitung des Befehls node zum
9          Zeitpunkt date. Der DOM-Teilbaum wird so wie hier übergeben an
10         das Modul übergeben, d.h. er muss der in das Element request
11         eingebettet sein. Als Rückgabe erhält der Aufrufer das Handle
12         des Eintrags, mit dem dieser Eintrag z.b. wieder gelöscht werden
13         kann. Für die korrekte Ausführung der Aktion ist das Speichern
14         des Handles nicht notwendig.
15         @param  date    Zeitpunkt der Ausführung
16         @param  node    DOM-Baum, der ausgeführt werden soll
17         @return         Handle des Eintrags
18     */
19     public int scheduleTask(java.util.Date date, org.w3c.dom.Node node);
20
21     /**
22         Diese Methode liefert die Liste der noch ausszuführenden Aufgaben
23         als Map zurück. Darin sind Elemente vom Typ SchedulerEntry
24         enthalten.
25         @return Liste der Tasks
26     */
27     public Map getTaskList();
28
29     /**
30         Diese Methode liefert die Liste der noch ausszuführenden Aufgaben
31         zwischen start und end als Map zurück. Darin sind Elemente vom Typ
32         SchedulerEntry enthalten.
33         @param  start   Zeitpunkt, nach dem die Tasks starten,
34                         null für unbegrenzt
35         @param  end     Zeitpunkt, bis zu dem die Tasks abgearbeitet sind,
36                         null für alle
37         @return Liste der Tasks
38     */
39     public Map getTaskList(java.util.Date start, java.util.Date end);
40
41     /**
42         Diese Methode löscht die Task mit der Nummer task. Diese Aufgabe
43         wird nicht abgearbeitet.
44     */
45     public void removeTask(int task);
46
47 }
```

C.9 Modulschnittstelle für Datenspeicher

DataStorage.java

```
1  package com.ewerk.dms.storage;
2
3  /** Dieses Interface dient zur Integration von Dokumentenspeichern. */
4  public interface DataStorage {
5      /** Diese Methode baut eine Verbindung zum Datenspeicher auf. Sie wird
6          stets zuerst aufgerufen. */
7      public void connect();
8
9      /** Diese Methode wird abschließend zu allen Aktionen aufgerufen */
10     public void disconnect();
11
12     /** Diese Methode gibt an, ob Transaktionen untertützt werden.
13         @return true, falls ja, sonst false
14     */
15     public boolean supportsTransactions();
16
17     /** Diese Methode startet eine neue Transaktion. Sind noch Transaktionen
18         offen, die nicht mit commit beendet wurden, so werden diese mit
19         einem Rollback beendet.
20         @return Transaktionsnummer
21     */
22     public int beginTransaction();
23
24     /** Diese Methode beendet eine Transaktion.
25         @param tid Transaktionsnummer
26         @param commit true, falls Änderungen gespeichert werden sollen.
27     */
28     public void endTransaction(int tid, boolean commit);
29
30     /** Diese Methode erstellt einen Block.
31         @param tid Transaktionsnummer
32         @return Blocknummer
33     */
34     public int createBlock(int tid);
35
36     /** Diese Methode lädt eine Block.
37         @param tid Transaktionsnummer
38         @param block Blocknummer
39         @return Block als Feld
40     */
41     public byte[] loadBlock(int tid, int block);
42
43     /** Diese Methode speichert einen Block. Vorhandene Blöcke werden ersetzt.
44         @param tid Transaktionsnummer
45         @param block Blocknummer
46         @param data Block als Feld
47     */
```

```
48      public void storeBlock(int tid, int block, byte[] data);
49
50      /** Diese Methode löscht einen Block.
51          @param tid Transaktionsnummer
52          @param block Blocknummer
53      */
54      public void deleteBlock(int tid, int block);
55
56      /** Diese Methode liefert eine Liste der vorhandenen Blocks als Feld.
57          @param tid Transaktionsnummer
58          @return Blockliste
59      */
60      public int[] listBlocks(int tid);
61  }
```

C.10 Modulschnittstelle für Eingabemodule

InputModule.java

```
1  package com.ewerk.dms.module;
2
3  import com.ewerk.dms.util.*;
4  import java.io.InputStream;
5  import java.util.*;
6
7  /**
8      Das Interface InputModule kapselt sämtliche Funktionen, die das DMS
9      verwendet, um Dateien des Benutzer in das DMS zu übertragen. Module,
10     die diese Schnittstelle implementieren, laden die Dokumente aus einem
11     durch den Benutzer zugreifbaren Bereich ab.
12     @see OutputModule
13 */
14 public interface InputModule {
15     /** Diese Methode liefert eine Parameterbeschreibung in Form eines
16         DOM-Baums zurück, der von der Client-Anwendung ausgewertet wird.
17         @return Parameterbeschreibung
18     */
19     public org.w3c.dom.Node getParameters();
20
21     /** Diese Methode wird zum Initialisieren der Parameter verwendet.
22         Sie wird vor allen anderen Aufrufen getätigt.
23         @param config Parameter nach Beschreibung von getParameters()
24     */
25     public void init(org.w3c.dom.Node config);
26
27     /** Diese Methode baut die Verbindung zum Datenspeicher des Benutzers
28         auf. Sie nach init() aufgerufen.
29     */
30     public void connect() throws ConnectionException, IOException;
31
32     /** Diese Methode wird zum Abschluss der Übertragung aufgerufen.
33     */
34     public void disconnect() throws ConnectionException, IOException;
35
36     /** Diese Methode liefert eine Liste der im Benutzerdatenspeicher
37         vorhandenen Datei und Verzeichnisse zurück.
38         @return Datei- und Verzeichnisliste
39     */
40     public Path[] getFileList() throws IOException;
41
42     /** Diese Methode liefert einen Eingabedatenstrom zum Lesen einer
43         bestimmten Datei zurück.
44         @return InputStream
45     */
46     public InputStream getFile(Path path) throws IOException;
47
```

```
48      /** Diese Methode liefert die Metadaten zu einer Datei oder einem
49          Verzeichnis zurück. Wird dies nicht unterstützt, so ist die
50          zurückgelieferte Map leer, jedoch nicht null.
51          @return Metadaten des Objekts
52      */
53      public Map getProperties(Path path) throws IOException;
54
55      /** Diese Methode zeigt an, ob das Modul die Übertragung von
56          Metadaten unterstützt.
57          @return true, falls ja, sonst false
58      */
59      public boolean canGetProperties();
60  }
```

C.11 Modulschnittstelle für Ausgabemodule

OutputModule.java

```
1  package com.ewerk.dms.module;
2
3  import com.ewerk.dms.util.*;
4  import java.io.InputStream;
5  import java.util.*;
6
7  /**
8      Das Interface OutputModule kapselt sämtliche Funktionen, die das DMS
9      verwendet, um Dateien zum Benutzer zu übertragen. Module, die diese
10     Schnittstelle implementieren, speichern die Dokumente in einem durch
11     den Benutzer zugreifbaren Bereich ab.
12     @see InputModule
13 */
14 public interface OutputModule {
15     /** Diese Methode liefert eine Parameterbeschreibung in Form eines
16         DOM-Baums zurück, der von der Client-Anwendung ausgewertet wird.
17         @return Parameterbeschreibung
18     */
19     public org.w3c.dom.Node getParameters();
20
21     /** Diese Methode wird zum Initialisieren der Parameter verwendet.
22         Sie wird vor allen anderen Aufrufen getätigt.
23         @param config Parameter nach Beschreibung von getParameters()
24     */
25     public void init(org.w3c.dom.Node config);
26
27     /** Diese Methode baut die Verbindung zum Datenspeicher des Benutzers
28         auf. Sie nach init() aufgerufen.
29     */
30     public void connect() throws ConnectionException, IOException;
31
32     /** Diese Methode wird zum Abschluss der Übertragung aufgerufen.
33     */
34     public void disconnect() throws ConnectionException, IOException;
35
36     /** Diese Methode legt ein Verzeichnis an.
37         @param path vollständiger Pfad des Verzeichnisses
38     */
39     public void putDirectory(Path path) throws IOException;
40
41     /** Diese Methode kopiert Daten in eine Datei.
42         @param path vollständiger Pfad der Zieldatei
43         @param is Eingabedatenstrom
44     */
45     public void putFile(Path path, InputStream is) throws IOException;
46
47     /** Diese Methode speichert die Metadaten zu einem Pfad ab. Dies kann
```

```
48          sowohl Verzeichnis als auch Datei sein.
49          @param path Pfad zum Objekt
50          @param properties Die Metdaten als Map
51      */
52      public void putProperties(Path path, Map properties) throws IOException;
53
54      /** Diese Methode zeigt an, ob das Modul die Übertragung von
55          Metadaten unterstützt.
56          @return true, falls ja, sonst false
57      */
58      public boolean canPutProperties();
59  }
```

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ort	Datum	Unterschrift