Personalisierungskonzepte für Digitale Bibliotheken

Diplomarbeit

Andre Zeitz
Universität Rostock
Fachbereich Informatik, LS DBIS
Albert-Einstein-Straße 21
D–18051 Rostock

Autor:	Andre Zeitz
Geburtstag:	20.10.1976
Geburtsort:	Gera
Matrikelnummer:	096201992
Betreuer:	Dr. Holger Meyer Dipl.–Inf. Mathias Bietz
Erstgutachter:	Prof. Dr. Andreas Heuer
Zweitgutachter:	Prof. Dr. Adelinde M. Uhrmacher
Bearbeitungszeitraum:	15.05.2001 – 14.11.2001
Abgabe:	14.11.2001

Zusammenfassung

Ein entscheidender Nachteil heutiger Digitaler Bibliotheken Lösungen ist ihr unpersönlicher Charakter. Nutzer einer Digitalen Bibliothek möchten sich aber beispielsweise ihre eigene Sicht auf den Datenbestand definieren, Dokumente annotieren oder informiert werden, wenn sich Änderungen an Datenbeständen ergeben, die für sie von Interesse sind.

In dieser Diplomarbeit werden darum Konzepte zur Personalisierung Digitaler Bibliotheken vorgestellt und in Form von Diensten in die Architektur einer bestehenden Digitalen Bibliothek eingefügt. Die Implementierung einiger dieser Dienste schließt die Arbeit ab.

Abstract

A disadvantage of today's digital library solutions is their unpersonal character. But for instance users of digital libraries would like to create their own view onto data, annotate documents or get informed if changes are made to such data, users are interested in.

This diploma thesis presents concepts for personalization of digital libraries. Theses concepts will be added as services into an existing digital library architecture. Finally, some of these services will be implemented.

CR-Klassifikation

H.2.3 Anfragesprachen

H.2.8 Datenbankanwendungen

H.3.3 Information Retrieval, Information Filtering

H.3.4 Nutzerprofile und Benachrichtigungsdienste

H.3.7 Digitale Bibliotheken

Schlüsselwörter

Digitale Bibliotheken, Personalisierung, Datenbanksysteme, Benachrichtigung

Inhalt

Kapitel 1

Einleitung

Digitale Bibliotheken sind komplexe Systeme, die oft als eine Art “Wunderwaffe” gegen die Probleme mit einer ständig wachsenden und schwer handhabbaren Flut von Daten gesehen werden. Sie sind in der Lage, große Datenmengen verschiedenster Art zu verwalten und auf Knopfdruck individuell bereitzustellen. In der Praxis sind Digitale Bibliotheken jedoch noch von diesem Status entfernt, insbesondere der individuelle Charakter fehlt. Ziel dieser Arbeit ist es nun, dieses Manko zu beheben, indem “Personalisierungskonzepte für Digitale Bibliotheken” erarbeitet werden.

1.1 Aufgabenstellung

Ein entscheidender Nachteil Digitaler Bibliotheken ist ihr fehlender persönlicher Charakter. Alle Nutzer einer solchen Bibliothek werden gleich behandelt, nämlich anonym. Ein Wunsch ist nun, dieses Manko zu beheben. Das Ziel dieser Diplomarbeit ist darum, Möglichkeiten zu konzipieren, mit denen Anwendern einer Digitalen Bibliothek ein Gesicht gegeben und ihren Anforderungen an sie Rechnung getragen wird.

Nutzer einer solchen Bibliothek möchten beispielsweise informiert werden, wenn sich Änderungen an Dokumentenbeständen ergeben, für die sie sich interessieren. Außerdem möchten sie sich ihre eigene Sicht auf den Datenbestand definieren, Dokumente annotieren oder eigene Dokumente verwalten und mit anderen Nutzern teilen. Auch die Nutzung klassischer Bibliotheksdienste, wie etwa die Katalogsuche, sollen über das System einer Digitalen Bibliothek möglich sein.

Das konkrete Ziel dieser Arbeit besteht nun darin, die Architektur einer Komponente zu konzipieren, die die o. g. Anforderungen realisiert. Diese Komponente, die als Personalisierungskomponente bezeichnet wird, soll dann in die Architektur einer bereits bestehenden Digitalen Bibliothek (BlueRose/BlueView, siehe Abschnitt 2.3) eingegliedert werden. Abschließend ist ein Teil dieser entwickelten Architektur prototypisch zu implementieren.

1.2 Aufbau der Arbeit

Der Aufbau dieser Arbeit gestaltet sich wie folgt:

Kapitel 2 gibt eine kurze Einführung in die Thematik Digitaler Bibliotheken und führt einige Beispiele für existierende Systeme auf.

Kapitel 3 stellt Konzepte vor, die von einer personalisierten Digitalen Bibliothek zu realisieren sind. Diese Personalisierungskonzepte gelten gleichzeitig als Anforderungen für die hier zu entwickelnde Architektur.

Kapitel 4 bildet den Kern dieser Arbeit, in dem einzelne Dienste konzipiert werden, durch deren Einsatz die Personalisierung von Bibliothekensystemen möglich wird.

Kapitel 5 gibt zunächst einen Überblick über Schnittstellen, Protokolle und Programmiertechniken, die zur Implementierung von Komponenten einer (personalisierten) Digitalen Bibliothek verwendet werden können. Im Anschluß daran werden die prototypische Implementierung einiger Dienste und deren Funktionsweise erläutert.

Kapitel 6 faßt die Arbeit zusammen gibt einen kurzen Ausblick auf zukünftige Arbeiten, die sich aufbauend auf dieser Diplomarbeit realisieren lassen.

Kapitel 2

Digitale Bibliotheken

Das folgende Kapitel soll eine kurze Einführung in die Thematik Digitaler Bibliotheken geben. Insbesondere soll dabei auf den Aufbau und die verschiedenen Dienste eingegangen werden, die für Digitale Bibliotheken charakteristisch sind.

2.1 Definition und Aufgaben Digitaler Bibliotheken

Die Definitionen von Digitalen Bibliotheken, die in der Literatur gefunden werden können, sind relativ vielfältig. Ein sehr weitreichende bietet [EF00], nach der eine Digitale Bibliothek ein System ist, daß zunächst Texte, Bilder, Animationen, Ton- und Videoaufnahmen in einer sogenannten elektronischen Bibliothek vorhält. Darüberhinaus muß ein solches System eine Vielzahl von Bibliotheksdiensten in einem ortsübergreifenden Verbund anbieten. Dieser ortsübergreifende Verbund wird auch als virtuelle Bibliothek bezeichnet. Außerdem müssen in einem Bibliothekensystem verschiedene Dienste integriert sein, die einen effizienten Zugriff auf die Daten des Systems ermöglichen. Ein solcher Zugriff soll über eine einheitliche Systemoberfläche gestattet sein. Letztlich sollte die "Systemintelligenz" des Gesamtsystems über die der damit verbundenen Teile hinausgehen.

In [Pae96] werden die grundlegenden Aufgaben einer Digitalen Bibliothek wie folgt charakterisiert:

Finden geeigneter Informationsquellen Digitale Bibliotheken umfassen üblicherweise eine große Zahl von Informationsquellen, beispielsweise Dokumentenserver. Dabei kann jede dieser Quellen vollkommen unterschiedliche Daten beinhalten. Um nun eine Nutzeranfrage an ein Bibliothekensystem beantworten zu können, ohne diese Anfrage an alle beteiligten Datenquellen zu schicken, müssen zunächst die gefunden werden, die diese Anfrage "möglichst gut" beantworten können.

Finden von Daten Wurden geeignete Informationsquellen zu Beantwortung einer Nutzeranfrage gefunden, so müssen im nächsten Schritt auf diesen Quellen die gesuchten Daten gefunden werden.

Auswerten der Daten Die auf den Datenquellen gefunden Daten sind u. U. sogenannte Rohdaten, die in dieser Form noch nicht verwendet werden können. In diesem Fall müssen die Daten vorverarbeitet werden.

Lokales Verwalten der Daten Alle auf verschiedenen Datenquellen gefundenen und vorausgewerteten Daten müssen im nächsten Schritt zusammengeführt und für die schnelle Bearbeitung zukünftiger Anfragen (lokal) gespeichert werden.

Bereitstellung der Daten Ein Digitales Bibliothekensystem hat immer mehrere Nutzer, die oftmals zusammen arbeiten. Aus diesem Grund ist es notwendig, alle Daten, die in vorigen Schritten gefunden, vorverarbeitet und zusammengeführt wurden, den verschiedenen Nutzern bereitzustellen.

2.2 Grundlegender Aufbau

Nach den Ausführungen im letzten Abschnitt läßt sich zunächst festhalten, daß eine Digitale Bibliothek aus verschiedenen Datenquellen besteht, also ein föderiertes System bildet. Als die wichtigsten Dienste stellen sich die folgenden heraus:

- Suche und Retrieval von Daten auf den beteiligten Datenquellen
- Replikation von gefunden Daten, etwa um eine gewisse Autonomie gegenüber den Datenquellen zu erreichen[1]
- Broker und Trader, die Kenntnis darüber besitzen, welche Datenquellen existieren und welche Art von Daten sie enthalten
- der Föderationsdienst, der die verschiedenen Datenquellen miteinander verknüpfen kann, da er beispielsweise in der Lage ist, die verschiedenen Datenformate und Protokolle der Quellen aufeinander abzubilden

Der Aufbau einer Digitalen Bibliothek könnte sich demnach wie in Abbildung 2.1 gestalten, der aus dem BlueRose-Projekt (siehe nächster Abschnitt) entnommen wurde. Dabei sind zwei Gruppen an einer solchen Bibliothek beteiligt: die Informationanbieter und die Informationsnutzer. In die Gruppe der Informationsanbieter fallen z. B. Verlage oder Bibliotheken, die ihre Daten, meist in Form von Dokumenten bereitstellen. Als Informationsnutzer gelten alle Benutzer eines solchen Bibliothekensystems.

Der Virtuelle Dokumentenserver (VDS) ist ein Verbund verschiedener lokaler Datenquellen, den sogenannten Lokalen Dokumentenservern (LDS). Nutzeranfragen werden immer an den Virtuellen Dokumentenserver gestellt, der diese einerseits an die Lokalen Dokumentenserver und andererseits über Föderationsdienste, Broker und Trader an entfernte Datenquellen verteilt. Die Anfrageergebnisse der einzelnen Datenquellen werden zum Schluß vom VDS gemischt und an den Nutzer zurückgegeben. Der Profil- und

[1] Fällt eine Datenquelle aus, so stehen ihre Daten trotzdem zur Verfügung, sofern sie vorher repliziert wurden.

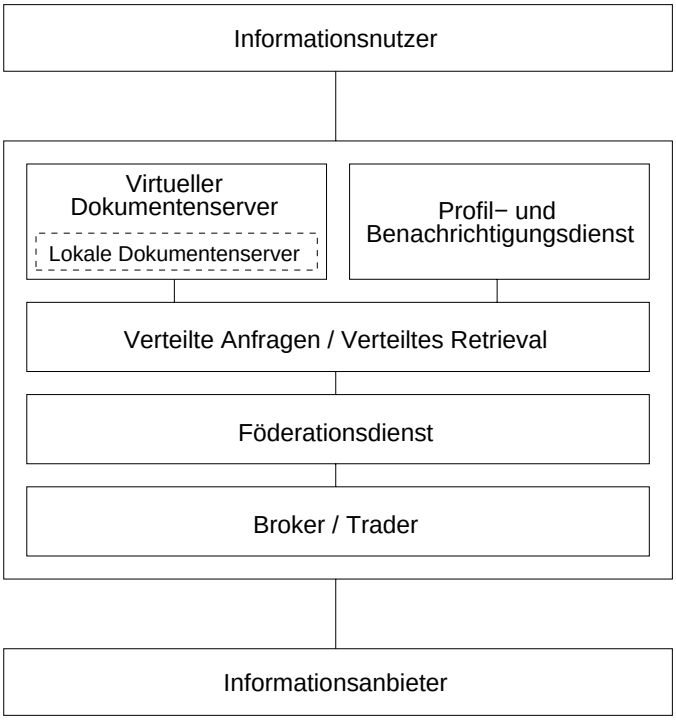

Abbildung 2.1: Grundsätzlicher Aufbau einer Digitalen Bibliothek

Benachrichtigungsdienst ermöglicht es Nutzern, ihre Interessen innerhalb von Profilen abzulegen, um sich dann informieren zu lassen, wenn den Datenquellen Ressourcen hinzugefügt wurden, die ihrem Interessenprofil entsprechen.

2.3 Konkrete Realisierungen

Inzwischen gibt es viele Digitale Bibliotheken, die z. T. auch für beliebige Nutzer offenstehen. Eine der bekanntesten davon ist die ACM Digital Library (*ww.acm.org/dl*), in der alle in den letzten Jahren bei der ACM veröffentlichten Artikel gespeichert sind. Der Zugriff auf diese Bibliothek erfolgt über eine Anfrage. Anfrageergebnisse können dann in sogenannten Book-Shelves (Bücherregalen) abgespeichert werden.

Eine andere Bibliothek ist der Link-Service des Springer-Verlages (*link.springer.de*), der wiederum alle Veröffentlichungen des Springer-Verlages vorhält. Zusätzlich wird noch ein Benachrichtigungsdienst betrieben, mit dem sich Nutzer informieren lassen können, sobald dem Bestand neue Publikationen hinzugefügt wurden.

An der Universität Rostock wurde mit den Projekten BlueRose (Building Libraries Unifying Enhanced Retrieval-Oriented User SErvices) und BlueView (BlueRose Virtual Integration of Electronic Web Services, [HMPT00]) ein Bibliothekensystem geschaffen, daß im Gegensatz zu den beiden o. g. Systemen mehrere Datenquellen in sich vereint. So wurden der Springer-Link-Service, die ACM DL und ein lokaler Dokumentenserver angeschlossen, dem Nutzer beliebige Dokumente hinzufügen können.

In [DDV+01] wird ein Projekt vorgestellt, das auch als Digitale Bibliothek betrachtet werden kann. Dabei handelt es sich um ein E-Commerce-System, daß seine Nutzer beobachtet, um daraus Rückschlüsse über ihre Interessen zu ziehen. Auf diese Weise besteht etwa die Möglichkeit, das Nutzerverhalten vorauszusagen (Prefetching) und die Aktionen, die der Nutzer wahrscheinlich als nächstes ausführen möchte, schon im Voraus auszuführen. Somit läßt sich für den Fall, daß die Voraussage korrekt war, eine schnellere Antwortzeit des Systems gegenüber dem Nutzer erreichen.

Das an der Universität Ulm erstellte System PADDLE (Personalizable Adaptable Digital Library Environment, [THK99]), ist nicht direkt im Bereich der Digitalen Bibliotheken anzusiedeln, sondern eher im Bereich der Personalisierung. Das Ziel von Paddle ist die individuelle Anpassung von Web-Dokumenten. In welcher Weise diese angepaßt werden sollen, wird dabei mit Hilfe von Metadaten festgelegt. Fordert nun ein Nutzer bzw. eine Applikation ein Web-Dokument an, dann erfolgt der Zugriff darauf nicht direkt, sondern über Paddle. Das System holt das geforderte Dokument, paßt es entsprechend den Metadaten an und gibt es schließlich zurück.

Alle diese Systeme bieten zumindest momentan nur eine unzureichende Umsetzung von Personalisierungskonzepten. Einige unterstützen zwar einzelne Konzepte wie die Benachrichtigung oder das Speichern von Interessen bzw. von Anfrageergebnissen, wünschenswert ist aber eine Architektur bzw. ein System, in dem möglichst viele solcher Konzepte realisiert werden.

Kapitel 3

Anforderungen an die Personalisierungskomponente

In Kapitel 2 wurden bereits die zentralen Anforderungen an Digitale Bibliotheken beschrieben. Diese sollen nun bezüglich der Personalisierung verfeinert werden. Dieses Kapitel beschreibt also die Anforderungen, denen eine Personalisierungskomponente in einer Digitalen Bibliothek gerecht werden sollte. Dabei werden zunächst grundsätzliche Anforderungen erläutert, die Digitalen Bibliotheken zugrunde liegen: Datenschutz und Skalierbarkeit. Im Anschluß daran werden alle Dienste eingeführt, die eine solche Komponente realisieren sollte, wobei hier zwischen primären und sekundären Diensten unterschieden wird. Als primär sollen die Dienste betrachtet werden, die zur Durchführung von Personalisierung als unbedingt notwendig erachtet werden. Sekundär sind die Dienste, die darüberhinaus angeboten werden können.

3.1 Datenschutz

Zur Durchführung von Personalisierung müssen Nutzerdaten, etwa Profile, und Ressourcen wie Dokumente, Annotationen etc. verwaltet werden, die mit Zugriffsrechten zu versehen sind. Dabei muß die Einhaltung dieser Rechte garantiert werden, d. h. in der Architektur der Digitalen Bibliothek und insbesondere der Personalisierungskomponente muß der Grundsatz des Schutzes von Daten verankert sein. Gründe dafür sind beispielsweise, daß Autoren, die ihre Dokumente in einer Digitalen Bibliothek verwalten, diese Dokumente vor dem Zugriff bestimmter Nutzer schützen möchten. Verlage, von denen der Großteil der Dokumente einer Digitalen Bibliothek stammt, verlangen die Einhaltung ihrer Lizenzbestimmungen, was u. a. wieder den Schutz der Dokumente vor unberechtigten Nutzern beinhaltet. Profile, die etwa die Interessen und persönliche Angaben der Nutzer, wie deren E-Mail-Adresse, enthalten, müssen außerdem vor dem unberechtigten Zugriff

geschützt werden. Der Datenschutz kann im Wesentlichen auf drei verschiedenen Ebenen erfolgen:

1. Durch kryptifizierte Speicherung der Daten.
2. Durch sichere Übertragung der Daten.
3. Durch Entkoppelung von allgemeinen und nutzerspezifischen Daten.

Bei der kryptifizierten Speicherung werden alle Daten immer zuerst verschlüsselt und dann gespeichert. Dementsprechend muß vor dem Auswerten der Daten ihre Entschlüsselung vorausgehen. Dieses Verfahren bewirkt, daß ein unberechtigter Nutzer keinen Zugriff auf gespeicherte Daten erhält. Zur Durchführung diese Verfahrens muß garantiert werden, daß nur bestimmte Softwarekomponenten kryptifizierte Daten entschlüsseln können, d. h. die Komponenten, die auf die verschlüsselten Daten zugreifen wollen, müssen sich authentifizieren. Ein Möglichkeit zur Authentifizierung und zur Ver- bzw. Entschlüsselung bieten die sogenannten asymmetrischen Kryptoverfahren, bei denen die Verschlüsselung mit einem öffentlichen, die Entschlüsselung mit einem geheimen Schlüssel durchgeführt wird. Ein Authentifizierung kann durch Hinzufügen einer Signatur mit dem geheimen Schlüssel des Absenders zusätzlich zur Kryptifizierung mit dem öffentlichen Schlüssel des Adressaten erfolgen.

Bei der sicheren Datenübertragung werden dagegen alle Daten, die über einen sogenannten unsicheren Kanal übertragen werden, derart verschlüsselt, daß nur der berechtigte Empfänger in der Lage ist, sie wieder zu entschlüsseln. Diese beiden Konzepte sind orthogonal zueinander. Durch Kombination beider kann ein hoher Datenschutz gewährleistet werden.

Eine andere Variante zur Gewährleistung von Datenschutz ist die Entkoppelung von allgemeinen und nutzerspezifischen Daten. Dabei werden entweder die zu speichernden bzw. zu übertragenden Daten anonymisiert, d. h. aus ihnen ist nicht ersichtlich, welchen Nutzer sie betreffen, oder sie werden pseudonymisiert. Bei der Pseudonymisierung werden die Daten, die einen Nutzer charakterisieren, verfremdet. Beispielsweise kann der Nutzername durch eine andere Zeichenkette ersetzt werden, aus der sich ohne Zusatzkenntnisse keine Rückschlüsse auf den ursprüngliche Nutzernamen ziehen lassen.

Für die Personalisierungskomponente spielen vor allem die sichere Übertragung und die Anonymisierung bzw. Pseudonymisierung eine wichtige Rolle. Da die Nutzerdaten in Datenbanksystemen gehalten werden und diese momentan nicht in der Lage sind, die zu verwaltenden Daten nur verschlüsselt abzulegen, ist es derzeit nicht möglich, Daten kryptifiziert zu speichern.

Unabhängig von der Art des Datenschutzes besteht die Notwendigkeit, Nutzer eindeutig identifizieren zu können. Nur so kann gewährleistet werden, daß Daten nur an die Nutzer gelangen, die zum Zugriff darauf berechtigt sind.

3.2 Skalierbarkeit

Eine Digitale Bibliothek ist ein System großer Dimensionen: es verwaltet riesige Datenmengen, die einer häufigen Änderung bezüglich des Datenbestandes[1] und bezüglich der

[1] Die Änderung des Datenbestandes einer Digitalen Bibliothek beinhaltet zumeist das Hinzufügen neuer Daten (INSERT), weniger das Ändern (UPDATE) oder Entfernen (DELETE) von Daten.

verwendeten Datenformate unterliegt. Außerdem nutzen eine große Anzahl von Anwendern ein solches System, insbesondere simultan, also gleichzeitig. Diese Tatsachen stellen im Grunde eine Herausforderung an ein solches Bibliothekensystem dar, da es in der Lage sein muß, den Anforderungen der massenhaften Datenverwaltung und der großen Zahl gleichzeitiger Nutzer gerecht zu werden.

Da die zu entwickelnde Personalisierungskomponente an oberster Stelle des Systems angesiedelt ist und jeder Nutzer, der das System kontaktiert, auf diese Komponente zugreift, ist die Leistungsfähigkeit des Bibliothekensystems in großem Maße von der Leistungsfähigkeit dieser Personalisierungskomponente abhängig. Sie muß daher nicht nur robust sondern auch skalierbar sein, da sie sich sonst zu einem Flaschenhals für das Gesamtsystem entwickeln kann. Dies bedeutet insbesondere, daß sie auch bei einer großen Anzahl von Nutzern korrekt und effizient arbeitet und kurze Antwortzeiten auf Nutzeranfragen gewährleistet.

3.3 Primäre Dienste

Die primären Dienste einer Digitalen Bibliothek sollen in den folgenden Abschnitten unter dem Gesichtspunkt der Personalisierung erläutert und die Anforderungen an sie dargelegt werden. Zu diesen primären Diensten zählen:

- Die Verwaltung von Nutzerprofilen
- Die Vorverarbeitung von Anfragen an das Bibliothekensystem
- Die Verwaltung von lokalen Dokumenten
- Das Benachrichtigen von Nutzern über Änderungen an Datenbeständen
- Das Annotieren von Dokumenten
- Der Austausch von Ressourcen wie Annotationen unter den Nutzern
- Die Integration von klassischen Bibliotheksdiensten

Die Reihenfolge, in der die Dienste diskutiert werden, hat dabei keine Bedeutung.

3.3.1 Verwaltung von Nutzerprofilen

Grundlage jeglicher Personalisierung ist das Verwalten und Verwenden nutzerspezifischer Daten. Die Personalisierungskomponente muß daher zunächst in der Lage sein, grundlegende Angaben wie Nutzernamen und E-Mail-Adressen in sogenannten Profilen zu verwalten. Weiterhin muß die Komponente alle Angaben verwalten, die zur Bereitstellung und Nutzung der Dienste notwendig sind, die in diesem Kapitel beschrieben werden. Dazu gehören beispielsweise das Alertingverhalten (Abschnitt 3.3.4), das implizite Suchverhalten für die Anfragevorverarbeitung (Abschnitt 3.3.2) oder die Interessen des Nutzers, die sein Forschungsgebiet oder ihn interessierende Zeitschriften beinhalten können. Die Angabe von Interessen an bestimmten Forschungsgebieten kann z.B. durch Nutzung von standardisierten Klassifikationssystemen erfolgen, siehe etwa [ACM98].

Zur Gewinnung der Nutzerdaten gibt es zwei Varianten: entweder gibt der Nutzer die entsprechenden Daten ein, oder das System beobachtet ihn, während er das Bibliothekensystem verwendet. Aufgrund dieser Beobachtungen zieht das System dann Rückschlüsse auf die Interessen dieses Nutzers. Die Kombination beider Verfahren ist ebenfalls denkbar, d. h. der Nutzer kann Präferenzen, die das System aufgrund von Beobachtungen gemacht hat, nachträglich anpassen oder er kann bestimmte Einstellungen vornehmen, die vom System aufgrund seiner Art der Benutzung gegebenenfalls modifiziert werden.

Mit Hilfe der Personalisierungskomponente muß das Bibliothekensystem außerdem in der Lage sein, Nutzerprofile zu importieren und zu exportieren. Diese Notwendigkeit ergibt sich zum einen aus der Tatsache, daß es verschiedene Digitale Bibliotheken gibt, die sich nicht koppeln lassen. Trotzdem möchte der Nutzer auf allen diesen Systemen mit den gleichen Präferenzen arbeiten. Zum anderen kann es der Wunsch des Nutzers sein, daß seine Präferenzen nicht zentral auf dem Systemserver, sondern auf seinem privaten Rechner gehalten werden. Dies bedeutet zwar einen erhöhten Kommunikationsaufwand zwischen dem Client des Nutzers und dem Bibliotheksserver. Die dezentrale Verwaltung von Daten führt jedoch auch zu einer Steigerung des Datenschutzes, sofern Datenhaltung und Kommunikation auf den in Abschnitt 3.1 beschriebenen Konzepten aufbauen.

3.3.2 Anfragevorverarbeitung

Der Zugriff auf ein Bibliothekensystem erfolgt mittels einer Anfrage. Eine solche Anfrage kann dabei die Suche über Metadaten oder auch das Retrieval von Dokumenten bedeuten. Da sich die Personalisierungskomponente an oberster Stelle befindet, nimmt sie zunächst alle Metadaten-Anfragen entgegen, Retrieval-Anfragen werden dagegen direkt an den Virtuellen Dokumentenserver (VDS) gesendet. Im Anschluß daran werden die Anfragen (Metadaten-Anfragen) vorverarbeitet, d. h. sie werden entweder erweitert oder so verändert, daß Teile davon durch neue Teile ersetzt werden.

Bei der Erweiterung der ursprünglichen Anfrage werden bei der Vorverarbeitung Nutzerinteressen eingeflochten, die sich in den Profilen befinden.[2] Dies kann beispielsweise bedeuten, daß die Anfrage auf eine Domäne eingeschränkt wird, auf der zu suchen ist, etwa Datenbankliteratur. Weiterhin kann die Anzahl der maximal zurückzuliefernden Anfrageergebnisse festgelegt oder die Formate zurückzugebender Dokumente eingeschränkt werden. Auch die Gewichtung von Anfragetermen, insbesondere bei der Volltextsuche, kann durchgeführt werden, ebenso das Bestimmen einer Rankingfunktion. Eine Änderung beispielsweise einer SQL-Anfrage auf diese Weise kann durch Hinzufügen von Bedingungen zur WHERE-Klausel erreicht werden.

Liegt eine Anfrage, die durch Substitution verändert werden soll, als Zeichenkette vor, so basiert diese Änderung auf Zeichenersetzung. Liegt die Anfrage in einer anderen Struktur vor, etwa als Anfragebaum, so müssen sogenannte Knoten des Baumes verändert bzw. durch neue ersetzt werden.

Im letzten Schritt der Vorverarbeitung wird die veränderte Anfrage an den Virtuellen Dokumentenserver weitergereicht, der sich im Gesamtsystem unter der Personalisierungskomponente befindet. Seine Aufgabe besteht in der Verarbeitung dieser Anfrage

[2] Die Durchführung einer solchen Anfragevorverarbeitung bedingt, daß Nutzerprofile im System verwaltet und interpretiert werden können (siehe Abschnitt 3.3.1).

bzw. Weiterleitung auf eventuelle externe Dienste und in der Rückgabe des Anfrageergebnis an die Personalisierungskomponente.

Der Effekt einer Anfragevorverarbeitung entspricht dem des Sichtenkonzeptes von Datenbanken. Während beim Zugriff auf die Sicht einer Datenbank beispielsweise bestimmte Attribute bzw. Attributwerte ausgeblendet werden, werden bei der Anfragevorverarbeitung bestimmte Dokumente oder bestimmte Metadaten dieser Dokumente ausgeblendet. So kann erreicht werden, daß sich im Anfrageergebnis nur solche Daten befinden, die für den Nutzer von Interesse sind.

3.3.3 Verwaltung lokaler Dokumente

Dokumente, die in Digitalen Bibliotheken verwaltet werden, stammen meist von großen, kommerziellen Informationsanbietern (etwa Verlagen). Durch den Einsatz einer Personalisierungskomponente soll es nun auch möglich sein, daß Nutzer ihre eigenen Dokumente mit Hilfe eines Bibliothekensystems verwalten können. Eigene Dokumente können dabei die Ressourcen sein, die entweder von entfernten Dokumentenservern repliziert wurden, etwa um zukünftige Zugriffe zu beschleunigen, oder aus anderen Quellen stammen, etwa den selber erstellten Dokumenten.

Zur Verwaltung lokaler Dokumente zählt zunächst die Möglichkeit, Dokumente einzufügen (INSERT), zu aktualisieren (UPDATE) und wieder zu löschen (DELETE). Weiterhin soll über der Menge der lokalen Dokumente implizit[3] und explizit gesucht werden können, sowohl über den Volltexten als auch über deren Metadaten. Dabei muß der Index, der zum Information-Retrieval auf den Volltexten verwendet wird, nicht nur beim Einfügen eines neuen Dokumentes in den Datenbestand erweitert, sondern auch bei einem Update bzw. dem Entfernen eines Dokumentes aus dem System angepaßt werden.

Darüberhinaus sollen für jedes Dokument Zugriffsrechte vergeben werden, durch die festgelegt wird, ob andere Nutzer Zugriff auf das Dokument erhalten sollen. Dadurch wird auch bestimmt, ob externe Virtuelle Dokumentenserver auf lokalen, persönlichen Dokumenten suchen dürfen. Zur Festlegung von Zugriffsrechten für Metadaten gibt es zwei Möglichkeiten: entweder wird nach dem Grundsatz verfahren, daß jegliche Metadaten frei zugänglich sind und keiner Autorisierung bedürfen, oder es werden genau wie bei Dokumenten Zugriffsrechte vergeben. Sollen für Metadaten Rechte vergeben werden, dann können diese explizit durch Nutzer vergeben oder implizit durch das System, so daß alle Metadaten dieselben Zugriffsrechte besitzen wie die Dokumente, die sie beschreiben.

Bei der hier entwickelten Komponente soll es der Entscheidung des Nutzers obliegen, ob für Metadaten von lokalen Dokumenten Rechte vergeben werden. Vergibt er keine expliziten Rechte, so wertet das System dies derart, daß alle Nutzer bzw. externen Systeme Zugriff darauf erhalten.

3.3.4 Benachrichtigung

Die Datenbestände einer Digitalen Bibliothek unterliegen häufigen Änderungen. Meist werden dem Bibliothekensystem neue Dokumente und deren Metadaten hinzugefügt.

[3] Dabei bedeutet implizit, daß bei jeglicher Suche über einer beliebigen Dokumentenmenge auch immer über den lokalen Dokumenten gesucht wird.

Aber auch Aktualisierungen und das Löschen bestehender Daten ist möglich. Der Nutzer soll deshalb die Möglichkeit erhalten, sich über die Änderungen der Datenbestände informieren zu lassen, was hier als Alerting oder Benachrichtigung bezeichnet wird. Dabei soll die Benachrichtigung bei Änderung aller Datenbestände möglich sein, zu denen Metadaten verfügbar sind, wozu insbesondere auch der lokale Dokumentenbestand aus Abschnitt 3.3.3 zählt.

Der erste Schritt des Alertings besteht darin, daß der Nutzer dem System gegenüber die Datenbestände angibt, die für ihn von Interesse sind (Subskribierung). Weiterhin gibt er an, in welcher Weise er über die Änderungen informiert werden möchte, beispielsweise per E-Mail oder bei Anmeldung am System. Zuletzt muß die Häufigkeit angegeben werden, in der das System über Änderungen des Datenbestandes informieren soll (etwa täglich oder wöchentlich).

Registriert die Personalisierungskomponente nun eine Änderung an einem Dokumentenbestand, für den ein Nutzer Interesse angemeldet hat, so informiert sie zum einen diesen Nutzer auf dem von ihm gewünschten Weg (z.B. via E-Mail) bzw. nach Ablauf einer gewünschten Frist (z.B. nach einer Woche). Zum anderen frischt diese Komponente den Bestand der Metadaten auf, d. h. wurde ein Dokument gelöscht, so werden die dazugehörigen Metadaten ebenfalls gelöscht, sofern dies nicht bereits von anderen Systemkomponenten durchgeführt wurde. Interessanter ist das Update der Metadaten jedoch, wenn neue Dokumente in einen externen Datenbestand eingefügt wurden, etwa bei einem Verlag. Werden die Metadaten dieser Dokumente, für die sich Nutzer subskribiert haben, in den lokalen Datenbestand übernommen, so spiegelt dieser Datenbestand nach einer gewissen Zeit die Interessen der Nutzer wieder. Der Vorteil dessen liegt darin, daß eine Suche über den lokalen Metadaten gute Ergebnisse liefert, da sie u. a. die Daten beinhalten, die für die Nutzer von Interesse sind. Dies bedeutet wiederum, daß es in vielen Fällen ausreichend ist, über den lokalen Metadaten zu suchen, wobei eine solche lokale Suche sehr viel schneller bearbeitet werden kann als eine auf entfernte Server ausgedehnte Suche.

3.3.5 Annotationen

Oft wäre es hilfreich, in schon gelesenen Dokumenten bestimmte Passagen schnell wiederzuerkennen, beispielsweise Abschnitte, die besonders wichtig sind oder Fehler enthalten. Aus diesem Grund soll über die Personalisierungskomponente die Möglichkeit geschaffen werden, Dokumente zu annotieren, d. h. mit kurzen, unformatierten Texten zu versehen. Dabei kann der Nutzer für seine Annotationen Rechte vergeben, mit denen festgelegt wird, welche weiteren Nutzer Zugriff darauf erhalten sollen. Dabei werden die Rechte für Dokumente und deren Annotationen getrennt vergeben. Somit kann ein Nutzer z.B. Zugriff auf ein Dokument erhalten, die Einsicht in die dazugehörigen Annotationen bleibt ihm jedoch verwehrt. Der umgekehrte Fall, d. h. Zugriff auf Annotationen und kein Zugriff auf das Dokument, soll allerdings ausgeschlossen werden.

Außerdem soll es möglich sein, Dokumente an den Nutzer zu liefern, in denen die dazugehörigen Annotationen bereits enthalten sind. Ein Beispiel hierfür ist das PDF-Format (Portable Document Format), daß einen Mechanismus enthält, mit dem Annotationen angezeigt und mit dem Dokument gedruckt bzw. vor dem Drucken ausgeblendet werden können. Schließlich soll es auch möglich sein, über der Menge der Dokumente zu suchen, die annotiert wurden, soweit dies mit den Zugriffsrechten der Annotationen bzw.

annotierten Dokumente vereinbar ist. D. h. ein Nutzer kann nur über annotierten Dokumenten suchen, wenn er die entsprechenden Zugriffsrechte besitzt.

3.3.6 Kooperation

Eine Digitale Bibliothek ist nicht nur ein System, in dem große Datenbestände organisiert und bereitgestellt werden, sondern daß auch von vielen Anwendern kontaktiert wird. Der Grundsatz, daß diese Anwender miteinander kooperieren wollen, soll sich in der Personalisierungskomponente wiederspiegeln.

Im Umfeld Digitaler Bibliotheken bedeutet Kooperation den Austausch persönlicher Ressourcen (etwa von Dokumenten, Metadaten oder Annotationen). Wie bereits in den vorigen Abschnitten beschrieben, können alle persönlichen Ressourcen mit Zugriffsrechten versehen werden, wobei die Gestaltung der Zugriffsrechte bewußt einfach gehalten ist. So kann lediglich entschieden werden, ob andere Nutzer Zugriff erhalten oder nicht. Erhält ein Anwender Zugriff, so kann er ausschließlich lesend auf diese Ressourcen zugreifen. Ein schreibender Zugriff ist nur auf eigene Ressourcen möglich.

3.3.7 Anbindung Klassischer Bibliotheksdienste

Traditionelle Bibliotheken bieten inzwischen verschiedene Dienste in digitaler Form an. So ist es beispielsweise möglich, Bücher oder Zeitschriften vorzubestellen, in einem Katalog nach ihnen zu suchen oder auch die sogenannte Fernleihe digital abzuwickeln. Überschreitet ein Leser die Ausleihfrist, so wird er via E-Mail daran erinnert.

Die Personalisierungskomponente soll nun den Zugriff auf solche Bibliotheksdienste kapseln, d. h. der Nutzer soll über das Digitale Bibliothekensystem Zugriff auf diese Dienste erhalten. Außerdem sollen Dienste angeboten werden, die auf denen der klassischen Bibliotheken aufbauen und diese erweitern. Damit kann sich einer Nutzer daran erinnern lassen, daß seine Leihfrist in wenigen Tagen erreicht wird, beispielsweise um online diese Frist zu verlängern.[4] Außerdem kann er sich darüber informieren lassen, daß der Bestand der Bibliothek erweitert wurde, etwa weil neue Bücher gekauft oder verteidigte Diplomarbeiten dem Bestand hinzugefügt wurden.

3.4 Sekundäre Dienste

Alle Dienste, die zusätzlich zu in den vorigen Abschnitten vorgestellten Diensten angeboten werden können, werden als sekundäre Dienste bezeichnet. Zu ihnen zählen

- die Nachbearbeitung von Anfrageergebnissen
- das Hinzufügen von externen Dokumentenservern durch den Nutzer
- die Erstellung von Handapparaten
- das wiederholte Stellen von Suchanfragen bzw. das inkrementelle Stellen von Anfragen.

[4] Sobald die Leihfrist erreicht oder überschritten wird, kann die Frist für ein ausgeliehenes Buch nicht mehr online verlängert werden.

Diese Dienste sollen nun in den folgenden Abschnitten charakterisiert werden.

3.4.1 Anfragenachbearbeitung

Beim Stellen einer Anfrage an eine Bibliothekensystem ist Nutzern z.T. nicht klar, wie selektiv diese Anfrage ist. Ist sie nicht ausreichend selektiv, dann enthält das Anfrageergebnis Dokumente, nach denen "eigentlich" nicht gesucht werden sollte. In solchen Fällen muß die Anfrage üblicherweise erneut vom Nutzer an das System gestellt werden, wobei diese neue Anfrage spezieller als die erste ist. Eine bessere Variante ist es, die Anfrage nicht neu sondern inkrementell zu stellen, also zu verfeinern, was durch Nutzung der Personalisierungskomponente ermöglicht werden soll. Bemerkt der Nutzer also anhand eines Anfrageergebnisses, daß sein Anfrage zu allgemein war, so kann er sie nachträglich und beliebig oft verfeinern.

Darüberhinaus soll der Nutzer mittels Relevance Feedback in der Lage sein, zu bewerten, welche Güte das Anfrageergebnis bezüglich seiner Anfrage hat. Das System legt diese Gütebewertung im Nutzerprofil ab und zieht sie bei der Bearbeitung zukünftiger Anfragen heran, um diese so zu verändern,[5] daß das Anfrageergebnis die Interessen des Nutzers wiederspiegelt. Eine Gewichtung der Ergebnisse kann bei der Nachbearbeitung einer Anfrage ebenfalls durchgeführt werden. Dadurch kann einerseits das Anfrageergebnis nach den Ranking-Ergebnissen geordnet, andererseits Teile des Anfrageergebnisses gelöscht werden, so daß dem Nutzer nur Ergebnisse zurückgeliefert werden, die für ihn interessant sind.

Zum Sachverhalt der personalisierten Anfragenachverarbeitung gehört auch das Aufbereiten der Anfrageergebnisse entsprechend den vorhandenen Nutzerpräferenzen. In diesen kann beispielsweise festgelegt sein, nach welchen Attributen der Metadaten das Suchergebnis geordnet werden soll, sofern dies nicht bereits bei der Anfragevorverarbeitung (Abschnitt 3.3.2) festgelegt wurde.[6]

Ein letzter Punkt ist das Setzen der Anfrageergebnisse (Layout). Dazu werden ebenfalls Nutzereinstellungen verwendet. Festgelegt werden können dabei allgemeine Darstellungsattribute wie Farben, Anzahl der präsentierten Ergebnisse pro angezeigter Ergebnisseite, Fonteigenschaften oder auch das Format der Dokumente, die an den Nutzer geliefert werden. Liegt ein Dokument nicht im gewünschten Format vor, so muß es entsprechend konvertiert werden, soweit dies möglich ist.

Abschließend gilt es festzulegen, daß sich Anfragevor- und Nachverarbeitung nicht widersprechen sollten. Es ist zwar technisch problemlos, beispielsweise ein Anfrageergebnis mehrmals zu sortieren, jedoch steht dies im Widerspruch zur anzustrebenden Performanz bzw. Skalierbarkeit aus Abschnitt 3.2.

[5] Siehe dazu Abschnitt 3.3.2 (Anfragevorverarbeitung).

[6] Das Festlegen von Sortierungsattributen bei der Anfragevorverarbeitung ist zu favorisieren, da das Bearbeiten einer Anfrage in einem Datenbanksystem eine gewisse Sortierung impliziert, d. h. im Falle einer nachträglichen Sortierung bei der Anfragenachbearbeitung würde das Suchergebnis ein zweites Mal sortiert werden.

3.4.2 Hinzufügen neuer Dokumentenserver

Nutzeranfragen an ein Bibliothekensystem werden üblicherweise über den Virtuellen Dokumentenserver an alle Lokalen Dokumentenserver und u.U. auch über Broker und Föderationsdienste an entfernte VDS weitergeleitet. Bei der Nutzung externer VDS entscheiden wählen die Broker zunächst die externen VDS aus, die für die Bearbeitung der Anfrage am besten geeignet sind, um die Anfrage schließlich an diese weiterzuleiten. Der Nutzer hat bei diesem Auswahlprozeß keinen direkten Einfluß.

Die zu konstruierende Personalisierungskomponente soll dieses Manko nun beheben. Dazu kann der Nutzer zunächst entfernte Dokumentenserver (VDS) angeben, auf die seine Anfragen automatisch ausgeweitet werden sollen. Weiterhin soll die Nutzer beim Stellen einer Anfrage an das System explizit festlegen können, an welche entfernten Dokumentenserver diese Anfrage zusätzlich gestellt werden soll.

3.4.3 Anwendung eines Sichtenkonzeptes

Bestimmte Nutzergruppen interessieren sich z.T. nicht für ganze Dokumente eines Bibliothekenbestandes sondern, nur für Teile davon, wobei mehrere solcher Teile miteinander in Beziehung stehen. Beispielsweise kann die Literatur eines Dozenten für eine Vorlesung, dem sogenannten Handapparat, aus mehreren Abschnitten, einigen Zitaten und einzelnen Abbildungen unterschiedlicher Bücher bestehen.

Solche Handapparate sind Sichten, wie sie im Datenbankbereich verwendet werden. Die einzelnen Sichten sind dabei materialisiert, da die Teile verschiedener Dokumente in einem neuen, physischen Dokument gespeichert werden. Auf diese Weise kann die Zugriffszeit auf einen Handapparat gesteigert werden, denn er muß nicht bei jedem Zugriff neu erstellt werden. Außerdem ändern Nutzer die Zusammenstellung eines Handapparates üblicherweise nur selten, so daß auch die Materialisierung nur selten geändert werden muß. Die Konsistenzwahrung des Dokumentes, etwa bezüglich der Abschnittsnumerierung kann im Zusammenhang mit Datenbanksichten als Aggregation betrachtet werden.

Die Personalisierungskomponente soll die Erstellung und Nutzung von Handapparaten unterstützen. Dazu müssen einerseits Teile (Kapitel, Abschnitte o.ä.) aus vorhandenen digitalen Dokumenten extrahiert werden können, um diese dann in einem neuen Dokument zu speichern. Das neu erstellte Dokument, daß den Handapparat symbolisiert, soll dabei bezüglich der verschiedenen extrahierten Teile anderer Dokumente in sich konsistent sein. Unter Konsistenz wird dabei z.B. die durchgehende Numerierung der Kapitel und Abschnitte verstanden, oder das Vorhandensein eines Inhaltsverzeichnisses, daß den Inhalt des Handapparates wiedergibt.

3.4.4 “Wiederholte Suche”

Häufig stellen Nutzer bestimmte Anfragen mehrmals. Einerseits, weil sie erneut nach einem Dokument suchen, daß sie bereits gefunden, inzwischen jedoch wieder gelöscht haben. Andererseits um zu überprüfen, ob es neue Dokumente gibt, die für sie von Interesse sind. Um solche Anfragen auf einfache Weise durchzuführen, soll die Personalisierungskomponente geeignete Konzepte umsetzen und bereitstellen.

Ein Konzept bzw. ein Dienst, mit dessen Hilfe sich Nutzer über neue Dokumente informieren lassen können, wurde mit dem Alerting bereits in Abschnitt 3.3.4 vorgestellt.

Weitere Konzepte sind das Speichern und Aufrufen von Anfragen und von Anfrageergebnissen. Beim Speichern einer Anfrage wird lediglich die Anfrage selbst gespeichert. Bei jedem Aufruf wird sie ausgeführt und das Anfrageergebnis dem Nutzer präsentiert. Hat sich der Dokumentenbestand geändert, über dem ein Anfrage gestellt wurde, so spiegelt sich dies im Anfrageergebnis wieder, da neu hinzugekommene Dokumente im Ergebnis erscheinen bzw. gelöschte fehlen, so sie auf das Anfrageergebnis passen. Im Gegensatz dazu wird beim Aufruf eines gespeicherten Anfrageergebnisses dem Nutzer das Resultat einer Anfrage zurückgeliefert, die in der Vergangenheit gestellt wurde. In diesem Resultat können sowohl Dokumente vorkommen, die bereits aus dem Dokumentenbestand entfernt wurden als auch Dokumente fehlen, die in der Zeit nach Stellen der Anfrage dem Bestand hinzugefügt wurden.

Eine Erweiterung des Konzeptes zum Speichern von Anfragen und Anfrageergebnissen ist das Bilden einer Differenz zwischen einem Anfrageergebnis und der dazugehörigen Anfrage. Das Ergebnis einer solchen Differenzbildung sind die Daten, die sich seit den beiden Zeitpunkten geändert haben, zu denen die Anfrage jeweils gestellt wurde. Dies können beispielsweise Dokumente oder Metadaten sein, die hinzugefügt oder gelöscht wurden.

Darüberhinaus soll das Suchen über den gespeicherten Anfragen und Anfrageergebnissen durch die zu entwerfende Komponente umgesetzt werden, vor allem um dem Nutzer bei einer großen Menge derartiger Anfragen und Anfrageergebnissen eine gewisse Übersicht zu bieten.

Kapitel 4

Architektur

Dieses Kapitel beschreibt den Aufbau der Personalisierungskomponente, mit deren Hilfe die Durchführung von Personalisierung in Digitalen Bibliotheken möglich wird. Dabei werden die einzelnen Bestandteile dieser Komponente erläutert und in einen Zusammenhang gebracht, wobei der Schwerpunkt bei der Verwaltung von Nutzerprofilen und der Benachrichtigung liegen wird. Grundlage dieses Kapitels ist die Annahme, daß alle Komponenten eines Digitalen Bibliothekensystems mit Ausnahme der Personalisierungskomponente bereits vorhanden sind. Insbesondere wird der Virtuellen Dokumentenserver angebunden, der in [Tit99] bzw. in [Jon99] entwickelt wurde.

Ein entscheidendes Merkmal der Personalisierungskomponente ist, daß ihrem Aufbau eine Dienstearchitektur zugrunde liegt, die oft auch Basis allgemeiner Digitaler Bibliotheken ist. Dies bedeutet, daß diese im folgenden vorgestellte Komponente aus eigenständigen Teilen aufgebaut ist, den sogenannten Diensten. Somit bleibt die Komponente einerseits überschaubar und wird andererseits sehr robust, denn der Ausfall eines Dienstes bedeutet hier nicht zwangsweise den Ausfall des gesamten Systems.

4.1 Einordnung in die Gesamtarchitektur

Die Personalisierungskomponente ist Bestandteil des Digitalen Bibliothekensystems, das aus den Projekten BlueRose und BlueView hervorgegangen ist (siehe Kapitel 2). Abbildung 4.1 veranschaulicht, an welcher Stelle des Systems diese Komponente eingegliedert wird: sie befindet sich unterhalb des Nutzersystems, das in der Abbildung durch den Informationsnutzer symbolisiert wird, und überhalb des Virtuellen Dokumentenservers. Der Profil- und Benachrichtigungsdienst, der in dieser Abbildung separat aufgeführt wird, soll nun Bestandteil der Personalisierungskomponente sein.

Der schematische Aufbau der Personalisierungskomponente ist in Abbildung 4.2 dargestellt. Zu den einzelnen Bestandteilen dieser Komponente, die im Weiteren auch Dien-

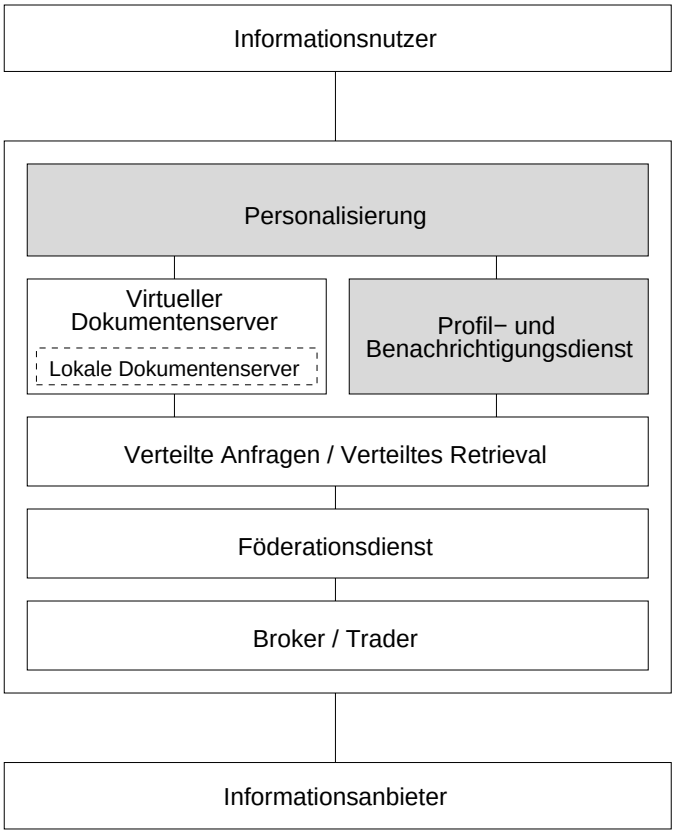

Abbildung 4.1: Gesamtarchitektur der Digitalen Bibliothek

ste genannt werden, zählen:

Benachrichtigungsdienst Dieser informiert Nutzer, wenn sich an Dokumentenbeständen Änderungen ergeben haben, die für sie von Interesse sind.

Anfrageverwaltung/Anfragedienst Der Anfragedienst führt ein Vor- bzw. Nachbearbeitung von Nutzeranfragen durch, während die Anfrageverwaltung diese Anfragen oder die dazugehörigen Anfrageergebnisse speichert.

Profildienst Der Profildienst verwaltet jegliche nutzerbezogenen Daten, etwa zur Konfiguration des Benachrichtigungs- oder Anfragedienstes.

Erweiterter Lokaler Dokumentenserver Der erweiterte LDS bietet dem Nutzer die Möglichkeit, eigene Dokumente mit Hilfe der Digitalen Bibliothek zu verwalten und anderen Nutzern zur Verfügung zu stellen.

Anbindung klassischer Bibliotheksdienste Neben den typischen Diensten einer Digitalen Bibliothek werden auch die Dienste einer klassischen Bibliothek, wie etwa die Katalogsuche oder die Fernleihe, an das System angebunden.

Annotationsdienst Der Annotationsdienst soll die Möglichkeit eröffnen, Randbemerkungen zu Dokumenten zu verfassen, zu speichern und mit anderen Nutzern auszutauschen.

Diese Dienste werden nun im Detail vorgestellt.

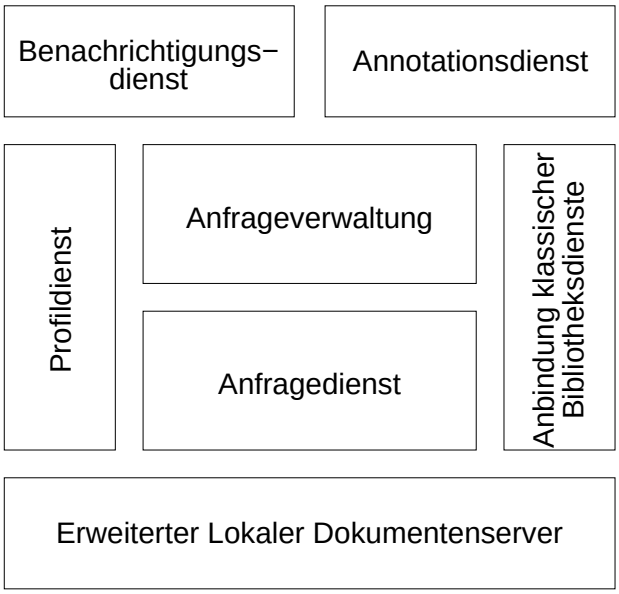

Abbildung 4.2: Grobarchitektur der Personalisierungskomponente

4.2 Primäre Dienste

In diesem Kapitel werden die primären Dienste einer Digitalen Bibliothek unter dem Fokus der Personalisierung konzipiert, zu denen, wie in Kapitel 3 beschrieben, die folgenden Dienste zählen:

- das Verwalten von Nutzerprofilen
- die Vorverarbeitung von Anfragen nach Metadaten
- die Verwaltung von lokalen Dokumenten
- die Benachrichtigung von Nutzern über Änderungen an Dokumentenbeständen
- die Möglichkeit, Dokumente zu annotieren
- eine Kooperationsmöglichkeit
- die Anbindung klassischer Bibliotheksdienste.

4.2.1 Verwaltung von Nutzerprofilen

Die Basis zur Realisierung einer Personalisierungskomponente ist das Verwalten und Benutzen von nutzerbezogenen Daten in sogenannten Profilen. Profile, die vom hier vorgestellten Profildienst verwaltet und bereitgestellt werden, werden in die folgendenen Gruppen unterteilt:

Nutzereigenschaften: Zu den Nutzereigenschaften zählen grundlegende Angaben des Nutzers, wie dessen vollständiger und sein Nutzername, sowie seine E-Mail-Adresse. Diese Eigenschaften werden einerseits benötigt, um Nutzer identifizieren und andererseits kontaktieren zu können, beispielsweise bei der Benachrichtigung. Die Eigenschaften des Nutzers werden üblicherweise nicht zu den Profilen gezählt, da sie allein noch nicht von großer Bedeutung sind und nicht bzw. selten verändert werden. Dennoch werden sie für die beiden folgenden Profiltypen benötigt, um diese den konkreten Nutzern zuordnen zu können.

Interessen: Die Interessen eines Nutzers werden durch Metadaten im Sinne des Dublin Core [WKLW98, DC99] beschrieben und können somit Autoren, Titel von Dokumenten u. s. w. sein. Dies können aber auch Metadaten des sehr viel mächtigeren Majour-Formates [Maj00] sein. Einzelne solcher Metadaten können logisch miteinander verknüpft werden. Alle Einträge innerhalb eines Interessenprofils werden konjunktiv verknüpft. Disjunktive Verknüpfungen können durch das Aufteilen von Interessen in verschiedene Interessenprofile erreicht werden. Negationen werden dagegen nicht realisiert.

Die Interessen eines Nutzers werden beispielsweise benötigt, um ihn über Änderungen an solchen Datenbeständen zu informieren, die für ihn relevant sind. Sie werden außerdem für Anfragevorverarbeitung benötigt, um eine vom Nutzer gestellte Anfrage zu konkretisieren oder um Anfrageergebnisse bei der Anfragenachbearbeitung einzuschränken. Nutzerinteressen können weiterhin Klassifikationen beinhalten, die auf bestimmten Klassifikationsschemata wie der CR-Klassifikation [ACM98] beruhen, mit der sich wissenschaftliche Literatur aus dem Bereich der Informatik beschreiben (klassifizieren) läßt. Zusätzlich zu einer Klassifikation muß immer ein Schwellwert angegeben werden, der die Ähnlichkeit zwischen der Klassifikation im Profil und der Klassifikation festlegt, die in den Alertingdaten angegeben wird. Nur wenn die Ähnlichkeit den Schwellwert erreicht, wird der Nutzer benachrichtigt.

Die Interessen werden in zwei verschiedenen Profilen gespeichert, den konkreten und den generalisierten Profilen. Während in den konkreten Profilen die genauen Interessen der Nutzer verzeichnet sind, beinhalten die generalisierten Profile die Daten, die durch Verallgemeinerung aus den konkreten Profilen hervorgehen, die sich “ähnlich” sind. Der Grund für diese Unterscheidung ist die Entlastung von Alerting-Servern, da auf diesen nur generalisierte Profile gespeichert und ausgewertet werden. Die Entlastung beruht darauf, daß die Anzahl der generalisierten Profile deutlich kleiner als die der konkreten Profile ist. Der hier verwendete Ähnlichkeitsbegriff wird im folgenden Abschnitt erläutert.

Einstellungen: Die Einstellungen beeinflussen das Systemverhalten bei der Kommunikation mit dem Nutzer. Sie umfassen beispielsweise die Dokumentenserver, auf denen vorrangig oder auch ausschließlich gesucht werden soll oder das Ranking von Suchergebnissen (Anfragevor- und Anfragenachbearbeitung). Auch das Alertingverhalten zählt zu dieser Gruppe der Profile, d. h. wie oft und in welcher Weise werden Nutzer über Änderungen an Dokumentenbeständen informiert.

Der Aufbau des Profildienstes ist in Abbildung 4.3 dargestellt. Sie enthält neben den verschiedenen Datenbasen, in den jeweils die verschiedenen Profiltypen verwaltet werden, einen Profilmanager. Dieser kapselt den Zugriff auf die Datenbasen und erzeugt aus den konkreten automatisch die generalisierten Profile. Er ist außerdem in der Lage, vollständige Profile zu exportieren und zu importieren. Ein vollständiges Profil umfaßt dabei alle Daten eines Nutzers, die in den verschiedenen Datenbasen gespeichert sind bzw. die beim Import dort abgelegt werden.

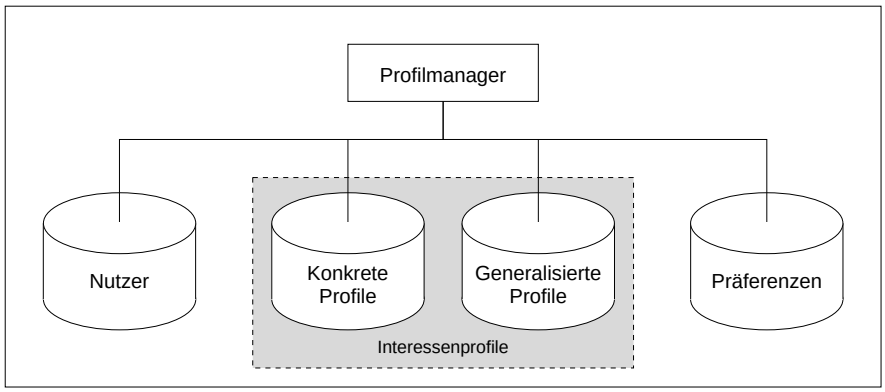

Abbildung 4.3: Aufbau des Profildienstes

Im folgenden sollen unter dem Begriff Profil jeweils die Interessen und die Einstellungen von Nutzern verstanden werden.

4.2.1.1 Ähnlichkeit von Klassifikationen

Interessenprofile bestehen, wie oben beschrieben, aus Metadaten, zu denen auch Klassifikationen gezählt werden können. Um solche Profile generalisieren zu können, muß zunächst anhand eines Maßes bestimmt werden, welche dieser Profile miteinander verallgemeinert werden. Ein derartiges Maß soll die Ähnlichkeit von Profilen sein, die durch einen skalaren Wert ausgedrückt wird. Profile, die sich sehr ähnlich sind, werden gemeinsam zu einem neuen Profil generalisiert.

Bevor hier eine Methode zur Bestimmung der Ähnlichkeit zweier Profile eingeführt wird, soll ein eigener Algorithmus vorgestellt werden, mit dem sich die Ähnlichkeit von Klassifikationen berechnen läßt. Diese Berechnung wird später Bestandteil des Verfahrens zur Berechnung der Ähnlichkeit von Profilen sein, soll aber getrennt davon vorgestellt werden, da sie sich von der Bestimmung der Ähnlichkeit von Metadaten unterscheidet.

Klassifikationssysteme wie die CR-Klassifikation [ACM98], die Mathematics Subject Classification [MSC] oder die Dewey-Klassifikation [DDC] sind hierarchisch aufgebaut und können auch als Menge von Bäumen betrachtet werden. Dabei befinden sich schwach klassifizierte Themen (Oberthemen) im oberen Teil eines Teilbaumes, stärker klassifizierte Themen (Unterthemen) befinden sich dagegen im unteren Teil. Ähnliche Themengebiete befinden sich dementsprechend im selben Teilbaum, Gebiete, die vollkommen verschieden voneinander sind, liegen in verschiedenen Teilbäumen. Die Bäume dieser Klassifikationssysteme haben alle eine Tiefe von drei, d. h. der Weg von der Wurzel zu einem beliebigen Blattknoten führt über drei Knoten. Diese Tiefe soll für den folgenden Algorithmus jedoch variabel sein, um auch Klassifikationen anderer hierarchischer Systeme damit vergleichen zu können.

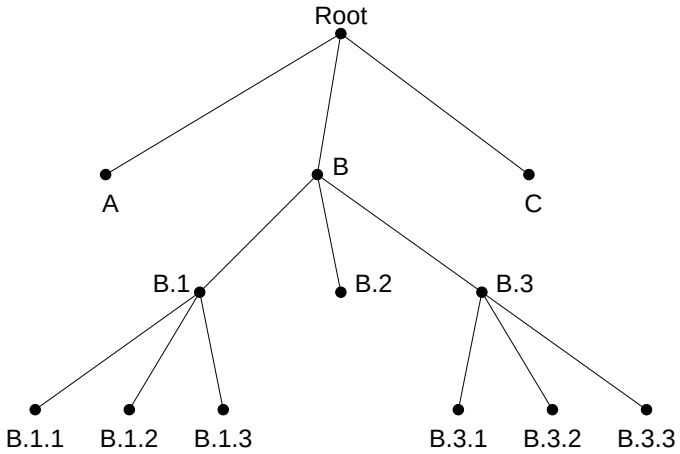

Abbildung 4.4: Ausschnitt eines Klassifikationsbaumes

Um den Algorithmus durchführen zu können, wird zunächst ein neuer Knoten eingeführt, der als Wurzel aller Teilbäume gilt. Auf diese Weise werden alle Bäume des Klassifikationssystems in einem Baum vereint, wie in Abbildung 4.4 dargestellt. Weiterhin werden die folgenden Größen festgelegt:

Symbol	Bedeutung
T	Tree, Klassifikationsbaum, Menge aus Knoten und Kanten
k	Maximale Tiefe des Klassifikationsbaumes
n	Node, Knoten des Klassifikationsbaumes
$d(n)$	Depth, Tiefe des Knotens n
$R(T)$	Root, Wurzelknoten des Klassifikationsbaumes für den gilt: $d(R(T)) = 0$
$L(T)$	Leafs, Menge der Blattknoten des Baumes, mit: $L(T) = \{n \mid d(n) = k\}$
$P(n)$	Parent, Vaterknoten von n, mit $d(P(n)) \leq d(n)$, d. h. ein Knoten kann sein eigener Vater sein

Der Algorithmus, der die Ähnlichkeit sim_C zweier Klassifikationen n_1 und n_2 berechnet und eine Abbildung auf das Einheitsintervall $[0,1] \in \mathbb{R}$ darstellt, gestaltet sich nun folgendermaßen:

Algorithmus 4.1 (Ähnlichkeit von Klassifikationen)

1. Finde den gemeinsamen Vater P der Knoten n_1 und n_2, der die maximale Tiefe besitzt.
2. Bestimme die Tiefe $d(P)$ dieses Vaterknotens.
3. Teile die Tiefe d durch die Tiefe des Knotens aus $\{n_1, n_2\}$, der sich tiefer im Baum befindet.

Dieser Algorithmus liefert für identische Klassifikationen immer den Wert 1, für Klassifikationen, die nur den Wurzelknoten als gemeinsamen Vater haben, den Wert 0. Die beiden folgenden Beispiele sollen zum Verständnis dieses Algorithmus beitragen.

Beispiel 4.1 Die Ähnlichkeit der Klassifikationen $B.1.3$ und $B.2.2$ berechnet sich wie folgt: der gemeinsame Vater maximaler Tiefe ist der Knoten B, der die Tiefe 1 besitzt. Die maximale Tiefe der beiden zu untersuchenden Knoten ist 3, so daß die Ähnlichkeit $sim_C(B.1.3, B.2.2) = \frac{1}{3}$ beträgt.

Beispiel 4.2 Zu berechnen ist die Ähnlichkeit der beiden Knoten $A.3.2$ und $A.3$. Der tiefste Vater beider ist $A.3$, dessen Tiefe 2 beträgt. Der Knoten $A.3.2$ befindet sich tiefer im Baum als $A.3$ und besitzt die Tiefe 3. Somit beträgt die Ähnlichkeit beider Klassifikationsknoten $\frac{2}{3}$.

4.2.1.2 Ähnlichkeit von Profilen

Die Bestimmung der Ähnlichkeit von Profilen wird zu Generalisierung von Profilen im Zusammenhang mit dem Benachrichtigungsdienst aus Abschnitt 4.2.4 benötigt. Sie ist daher nur für die Nutzerinteressen sinnvoll, da die Benachrichtigung von Nutzern auf den Interessen beruhen, die in ihren Interessenprofilen festgelegt sind. Wie schon oben beschrieben, bestehen solche Profile aus Metadaten, oder genauer aus Attributwert-Attributtyp-Paaren, wobei der Typ eines Attributs aus der Menge der Metadatentypen ist, wie etwa der Autor oder der Verlag eines Dokumentes. Sie können außerdem auch Klassifikationen enthalten.

Um Profile generalisieren zu können, muß zunächst festgestellt werden, welche der Profile zu einem neuen zu verallgemeinern sind. Dazu wird zuerst die Ähnlichkeit von Profilen bestimmt. Profile, die sich “zu einem gewissen Grad” ähnlich sind, werden dann nach dem in Abschnitt 4.2.1.3 vorgestellten Algorithmus generalisiert. Zu Berechnung der Ähnlichkeit von Profilen soll hier ein einfaches Ähnlichkeitsmaß vorgestellt werden, daß im Rahmen dieser Diplomarbeit entwickelt wurde.

Das im folgenden eingeführte Ähnlichkeitsmaß zweier Profile ist wieder eine Abbildung auf das Einheitsintervall $[0,1] \in \mathbb{R}$. Dazu werden zunächst die folgenden Festlegungen getroffen:

Symbol	Bedeutung
a	Attribut
T	Types, Menge der Typen $\{Creator, Title, Publisher, \ldots, Classification\}$
$t(a)$	Type, Typ eines Attributs a
$v(a)$	Value, Wert eines Attributs a
P	Profil, Menge von Attributen: $\{a \mid a\, ist\, Attribut\}$
$\lvert P\rvert$	Anzahl der Attribute des Profils P
$SUBSTR(v(a))$	Menge der Substrings vom Wert des Attributs a
X	Menge von Attributpaaren, wobei die Attribute aus jeweils verschiedenen Profilen stammen, vom selben Typ aber verschieden von *Classification* sind, denselben Wert besitzen und jedes Attribut höchstens einmal in der Menge vorkommen darf: $\{(a,b) : a \in P_1 \wedge b \in P_2 \wedge P_1 \neq P_2 \wedge t(a) = t(b) \neq Classification \wedge v(a) = v(b) \wedge (\nexists (c,d) \in X : c = a \vee d = b)\}$
Y	Menge von Attributpaaren, wobei die Attribute aus jeweils verschiedenen Profilen stammen, vom selben Typ aber nicht vom Typ *Classification* sein, ein Attributwert Substring des anderen ist und jedes Attribut höchstens einmal in der Menge vorkommen darf: $\{(a,b) : a \in P_1 \wedge b \in P_2 \wedge P_1 \neq P_2 \wedge t(a) = t(b) \neq Classification \wedge (v(a) \in SUBSTR(v(b)) \vee v(b) \in SUBSTR(v(a))) \wedge (\nexists (c,d) \in X : c = a \vee d = b)\}$
Z	Menge von Attributpaaren, wobei die Attribute aus jeweils verschiedenen Profilen stammen, vom Typ *Classification* sind, jedes Attribut höchstens einmal in der Menge vorkommen darf und die Ähnlichkeit sim_C dieser Klassifikationen maximal ist

Unter der Annahme, daß jedes Attribut nur in eine der Mengen aus $\{X, Y, Z\}$ aufgenommen werden darf, wird die Ähnlichkeit sim_P zweier Profile P_1 und P_2 wird nun wie folgt bestimmt:

$$sim_P(P_1, P_2) = \frac{|X| + 0.5|Y| + \sum_{i=1}^{|Z|} sim_C(a_i, b_i)}{\max\{|P_1|, |P_2|\}} \qquad \text{mit } (a_i, b_i) \in Z$$

Dabei wird im Zähler die Anzahl der Attributpaare aufsummiert, die vom selben Typ und deren Wert entweder identisch ist, oder der Wert eines Attributs Teilstring das anderen ist. Ist ein Attributwert Substring eines anderen, so wird dies mit dem Faktor 0.5 gewichtet, da die Attribute in diesem Fall nicht identisch sind, sondern nur ähnlich. Dazu werden die Ähnlichkeiten der Paare aus Z hinzuaddiert, wobei diese Ähnlichkeiten wie in Abschnitt 4.2.1.1 beschrieben, berechnet werden. Um das Ähnlichkeitsmaß auf das Einheitsintervall abzubilden, also eine Normierung durchzuführen, wird die oben berechnete Summe

durch die Anzahl der Attribute des Profils dividiert, das die meisten Attribute enthält. Auf diese Weise ergibt sich zwischen identischen Profilen eine Ähnlichkeit von 1 und zwischen Profilen, die vollkommen verschieden voneinander sind, eine Ähnlichkeit von 0. Das folgende Beispiel soll dies verdeutlichen.

Beispiel 4.3 Gegeben seien die beiden Profile P_1 und P_2, mit:

P_1

Attribut	Typ	Wert
a_1	Creator	Andreas Heuer
a_2	Title	Datenbanken
a_3	Publisher	Addison Wesley
a_4	Date	1997
a_5	Classification	H.2.8

P_2

Attribut	Typ	Wert
b_1	Creator	Andreas Heuer
b_2	Title	OO Datenbanken
b_3	Title	Datenbanken
b_4	Date	97
b_5	Contributor	Schlegelmilch
b_6	Classification	H.2

Da die Attribute a_1 und b_1 sowie a_2 und b_3 vom gleichen Typ sind und denselben Wert aufweisen, ergibt sich die Menge X wie folgt:

$$X = \{(a_1, b_1), (a_2, b_3)\}$$

Aufgrund der Tatsache, daß der Attributwert von a_2 ein Teilstring von b_2 sowie der Attributwert von b_4 Teilstring von a_4 ist, ergibt sich die Menge Y aus

$$Y = \{(a_2, b_2), (a_4, b_4)\}.$$

Die Menge Z enthält nur das Element (a_5, b_6). Die Ähnlichkeit dieser beiden Elemente beträgt $\frac{2}{3}$, berechnet nach dem Algorithmus aus Abschnitt 4.2.1.1. Die Mächtigkeit der Menge X ist 2, ebenso die der Menge Y. Das Profil P_2 enthält mit sechs Attributen mehr als das Profil P_1. Die Ähnlichkeit der beiden Profile berechnet sich nun aus

$$\begin{aligned} sim_P(P_1, P_2) &= \frac{\left|\left\{(a_1, b_1), (a_2, b_3)\right\}\right| + 0.5 \cdot \left|\left\{(a_2, b_2), (a_4, b_4)\right\}\right| + sim_C(a_5, b_6)}{\max\left\{|P_1|, |P_2|\right\}} \\ &= \frac{2 + 0.5 \cdot 2 + \frac{2}{3}}{6} \\ &= \frac{11}{18} \approx 0.61 \end{aligned}$$

Abschließend bleibt zu bemerken, daß dieses Ähnlichkeitsmaß sehr einfach ist, weil es u. a. von der Ähnlichkeit zweier Attributwerte ausgeht, sobald einer dieser Attributwerte ein Teilstring des anderen ist. Die Tatsache, daß beispielsweise ein Datum in vielen verschiedenen Formaten angegeben werden kann, widerspricht dieser Annahme. Enthält etwa ein Attribut eines Profils den Wert “7” stellvertretend für das Jahr 2007, und enthält das Attribut eines anderen Profils den Wert “2000-07-10” stellvertretend für den 10. Juli 2000, dann würde das oben beschriebene Ähnlichkeitsmaß eine Ähnlichkeit zwischen beiden Profilen erkennen, obwohl die Jahre 2000 und 2007 weit auseinander liegen. Eine Verbesserung des Maßes wäre beispielsweise das Heranziehen eines Kriteriums, das stärker als das Substringkriterium ist oder auch eine Gewichtung, die von den Attributtypen abhängt.

4.2.1.3 Generalisierung von Profilen

Voraussetzung für das Generalisieren von Profilen ist ein Ähnlichkeits- bzw. ein Abstandsmaß, mit dem sich bestimmen läßt, wie ähnlich bzw. wie weit voneinander entfernt Profile sind. Der Algorithmus, der hier vorgestellt wird, basiert auf dem Ähnlichkeitsmaß des letzten Abschnitts und wurde ebenfalls im Rahmen dieser Arbeit entwickelt. Zum besseren Verständnis und zur Übersichtlichkeit werden dazu die folgenden Symbole eingeführt:

Symbol	Bedeutung
k	Schwellwert, mit $0 \leq k \leq 1$ und $k \in \mathbb{R}$
S_i	Special – konkretes, zu generalisierendes Profil
G_j	General – allgemeines, generalisiertes Profil

Ausgehend von einer Menge konkreter Profile, gestaltet sich der Algorithmus wie folgt:

Algorithmus 4.2 (Ähnlichkeit von Interessenprofilen)

1. Existiert bereits ein G_j, so fahre mit (3) fort.

2. Die Menge der generalisierten Profile wird mit dem ersten konkreten Profil initialisiert: $G_1 := S_1$. Daraufhin wird G_1 wie folgt bereinigt:

   a) Existieren innerhalb dieses Profils Attribute a_m und a_n, die vom Typ *Classification* sind und bei denen a_m ein Oberthema von a_n darstellt, so entferne a_n.

   b) Existieren innerhalb dieses Profils Attribute a_m und a_n, die vom selben Typ sind und bei denen der Wert von a_m Substring des Wertes von a_n ist, dann entferne a_n.

   c) Entferne alle Attributduplikate.[1]

3. Berechne die Ähnlichkeit zwischen dem nächsten S_i und allen G_j. Existieren verschiedene G_j, zu denen die Ähnlichkeit größer oder gleich k ist, so füge S_i, wie unter (5) beschrieben, einem G_j hinzu, zu dem die Ähnlichkeit maximal ist.

4. Existiert kein G_j, mit $sim_P(S_i, G_j) \geq k$, dann ergibt sich ein neues G_k aus dem S_i. Fahre mit (6) fort.

5. Das Hinzufügen eines konkreten Profils S_i zu einem generalisierten Profil G_j geschieht attributweise wie folgt:

   a) Entferne alle Attribute aus G_j, deren Typ nicht in S_i vorkommt.

   b) Entferne alle Attribute a_m vom Typ *Classification* aus G_j, zu denen es kein Attribut a_n gleichen Typs aus S_i gibt mit $sim_C(a_m, a_n) \geq k$.

   c) Entferne alle Attribute a_m aus G_j, zu denen es kein Attribut a_n gleichen Typs aus S_i gibt mit $v(a_m) \in SUBSTR(v(a_n))$ oder $v(a_n) \in SUBSTR(v(a_m))$.

   d) Ist das hinzuzufügende Attribut a_n vom Typ *Classification* und existieren innerhalb von G_j Klassifikationsattribute, zu denen die Ähnlichkeit größer oder gleich k ist, dann wähle das Attribut a_m aus G_j, zu dem die Ähnlichkeit maximal ist. Ist a_n ein Oberthema von a_m, dann ersetze a_m durch a_n.

   e) Enthält S_i Attribute $\{a_1, \ldots, a_e\}$, deren Wert Substring vom Wert eines Attributs a_m aus G_j ist, wobei die Typen dieser Attribute identisch sind, dann entferne a_m aus G_j und füge alle Attribute aus $\{a_1, \ldots, a_e\}$ in das Profil G_j ein.

6. Speichere in S_i ab, in welches G_j es durch die Generalisierung eingegangen ist und in G_j, aus welchen S_i es entstanden ist.

7. Fahre mit (3) fort, bis alle Profile generalisiert wurden.

Das Ergebnis dieses Algorithmus ist eine Menge von allgemeinen Profilen, die die gegebenen konkreten Profile subsumieren, d. h. die Menge der generalisierten Profile überdeckt die gesamten Daten der konkreten Profile. Somit sind alle Interessen, die in den

[1] Ein Attribut ist genau dann Duplikat eines anderen, wenn es den selben Typ und den gleichen Wert besitzt.

konkreten Profilen angegeben sind, auch in den allgemeinen Profilen enthalten. Die Anzahl der allgemeinen Profile, die dieser Algorithmus erzeugt, hängt wesentlich von der Wahl des Schwellwertes k ab. Wird dieser Wert sehr klein gewählt, dann entstehen sehr wenige allgemeine Profile und umgekehrt. Für $k = 0$ entsteht genau ein generalisiertes Profil, das unter Umständen leer ist und somit auf alle Interessen paßt. Bei $k = 1$ werden nur identische Profile zusammengefaßt, alle anderen werden einfach in die Menge der allgemeinen Profile übernommen. Ein Beispiel, mit dem dieser Algorithmus an einigen Profilen vorgeführt wird, ist in Anhang A.1 zu finden.

Wird durch die Änderung der Interessen eines Nutzers sein Interessenprofil verändert, so muß auch das Profil angepaßt werden, daß aus der Generalisierung dieses Interessenprofils hervorgegangen ist. Dies kann beispielsweise durchgeführt werden, indem beim Generalisieren abgespeichert wird, welches Attribut welches konkreten Profils in welcher Weise in das allgemeine Profil eingegangen ist, um anhand dieser Informationen das generalisierte Profil anzupassen. Dieser Weg ist jedoch aufwendig und ungleich komplizierter als der hier vorgeschlagene. Dabei wird einfach das allgemeine Profil gelöscht, in welches das geänderte Profil durch Generalisierung eingeflossen ist. Anschließend werden alle Profile, die an der Generalisierung dieses allgemeinen Profils beteiligt waren, erneut generalisiert. Der Aufwand dafür hängt maßgeblich von der Anzahl der neu zu generalisierenden Profile ab, welche wie oben beschrieben durch die Wahl des Schwellwertes k beeinflußt wird.

Löscht ein Nutzer sein Interessenprofil, dann wird, wie beim Ändern eines Interessenprofils, das dazugehörige allgemeine Profil gelöscht. Alle anderen konkreten Profile, die an der Generalisierung dieses allgemeinen Profils beteiligt waren, erneut generalisiert.

Eine vollkommen andere Variante, um auf Änderungen der Nutzerprofile zu reagieren, besteht darin, die dazugehörigen generalisierten Profile zunächst unverändert zu lassen, und lediglich das veränderte Interessenprofil neu zu generalisieren. Der Vorteil dieses Verfahrens ist seine Einfachheit und seine Performanz, da insbesondere beim Löschen eines konkreten Profils das dazugehörige allgemeine Profil unangetastet bleibt. Damit geht jedoch der entscheidende Nachteil einher, der darin besteht, daß das System von Benachrichtigungsdiensten u. U. über Änderungen an Datenbeständen informiert wird, die für die Nutzer nicht mehr von Interesse sind. Das System wird also häufiger informiert als notwendig wäre, was zu einem zusätzlichen Aufwand zur Auswertung von Alerting-Daten führt.

Abschließend ist bezüglich des vorgestellten Algorithmus zu erwähnen, daß er starke Ähnlichkeit zu Clustering-Algorithmen aus dem Data Mining-Bereich (siehe etwa [HK00]) aufweist, bei denen Punkte im n-dimensionalen Raum in Klassen eingeteilt werden, die untereinander meist disjunkt sind. Der entscheidende Unterschied zu den Clustering-Algorithmen besteht jedoch darin, daß die zu generalisierenden Profile beliebig viele Attribute besitzen können, während ein zu klassifizierender Punkt immer eine bestimmte Dimension aufweist. Außerdem entsteht durch den hier vorgestellten Algorithmus keine Wolke sondern nur genau ein Element, das generalisierte Profil, was beim Clustering etwa dem Durchschnitt der Punktabstände innerhalb einer Wolke entspricht. Der vorgestellte Algorithmus weist außerdem eine gewisse Ähnlichkeit mit dem Algorithmus zum Aufbau von Versionsräumen (siehe z. B. [RN95]) auf. Ein Versionsraum wird durch das Hinzufügen von positiven und negativen Trainingsbeispielen aufgebaut, wobei sich dessen obere (allgemeine) und untere (spezielle) Grenze verändern. Beim Generalisieren

von Profilen entsprechen die konkreten Profile den Trainingsbeispielen. Das generalisierte Profil, das dem Versionsraum entspricht, wird beim Ersetzen eines neuen allgemeineren Attributwertes allgemeiner, was dem Verschieben der allgemeinen Grenze im Versionsraum verglichen werden kann. Durch das Hinzufügen eines neuen Attributs zum generalisierten Profil wird dieses dagegen konkretisiert, was wiederum dem Verschieben der speziellen Grenze im Versionsraum entspricht. Der eigentliche Unterschied besteht im eigentlichen Ziel beider Algorithmen: Im Versionsraum bewegen sich die Grenzen aufeinander zu, um im Idealfall miteinander zu verschmelzen. Wie allgemein oder speziell dagegen ein verallgemeinertes Profil ist, spielt keine Rolle. Im Vordergrund steht nur das Erzeugen eines Profils, das andere Profile wiederspiegelt, also subsumiert.

4.2.1.4 Management von Profilen

Zum Management der Profile, die durch die Profilverwaltung der Personalisierungskomponente verwaltet und bereitgestellt werden, gehört neben dem Einfügen, Ändern und Löschen sowohl kompletter Profile als auch Teilen davon (Interessenprofile, Einstellungen) auch das automatische Generalisieren von neu hinzugefügten Profilen. Hinzu kommt, daß Änderungen von konkreten Profilen auf die dazugehörigen generalisierten Profile übertragen werden. Dabei muß immer festgehalten werden, welche konkreten Profile an der Erstellung eines allgemeinen Profils beteiligt waren. Nach dem Erstellen eines generalisierten Profils und nach dessen Änderung ist dieses Profil an den Benachrichtigungsdienst (Abschnitt 4.2.4) zu senden. Nach dem Löschen eines generalisierten Profils im Profildienst ist es ebenfalls im Benachrichtigungsdienst zu entfernen.

Eine besondere Behandlung von Profilen ist dann gefordert, wenn diese nicht dauerhaft im System, sondern extern gespeichert werden sollen, etwa auf einem Nutzeraccount. In diesem Fall werden bei der Systemanmeldung des Nutzers dessen Profile in das System importiert, bei Systemabmeldung wieder exportiert und im System durch Profile ersetzt, aus denen lediglich hervorgeht, daß die eigentlichen Profile extern vorliegen und welchem Nutzer sie gehören. Das Anlegen eines solchen Profils geschieht genauso wie das Anlegen von im System gespeicherten Profilen, was insbesondere bedeutet, daß es generalisiert wird. Die gesonderte Behandlung von extern gespeicherten Profilen bei der Benachrichtigung wird im folgenden und den Abschnitten 4.2.4.3 bis 4.2.4.6 beschrieben.

4.2.1.5 Import und Export von Profilen

Eine wichtige Fähigkeit der Profilverwaltung sind der Im- und Export von Nutzerprofilen. Dies ergibt sich zum einen aus der Tatsache, daß mehrere Digitale Bibliothekensysteme existieren, die sich nicht miteinander koppeln lassen, etwa aus lizenzrechtlichen Gründen. Trotzdem soll es Nutzern möglich sein, die verschiedenen Systeme mit denselben Profilen zu benutzen. Zum anderen sollen Nutzer beispielsweise aus Gründen des Datenschutzes eigene Profile außerhalb des Systems speichern können. Auf welche Weise der Nutzer in diesem Fall seine Profile verwaltet, z. B. im Datei- oder in einem Datenbanksystem, ist hierbei nicht von Bedeutung. Entscheidend ist nur, daß das Bibliothekensystem die so gespeicherten Profile importieren und sie bei Änderung überschreiben kann.

Der Import und der Export von Nutzerprofilen geschieht immer mit kompletten Profilen, wobei ein komplettes Profil aus der Verknüpfung von allen Nutzerdaten, Interessen-

profilen und Einstellungen besteht. Dabei kann der Im- bzw. Export eines solchen kompletten Profils entweder direkt durch den Nutzer initiiert werden, oder indirekt, indem ein Nutzer seine Profile nicht im Bibliothekensystem verwaltet, sondern extern. Dabei werden, wie bereits beschrieben, die extern gespeicherten Profile, die sich beispielsweise auf dem Rechner des Nutzers befinden können, bei der Systemanmeldung des Nutzers vom System importiert und benutzt. Bei der Abmeldung vom System werden sie bei Änderung exportiert, aus dem System entfernt und durch Profile ersetzt, aus denen lediglich hervorgeht, daß die “eigentlichen” Profile extern vorliegen und welchem Nutzer sie gehören.

Das hier zum Austausch der Profile verwendete Format ist XML [BPSMM00]. Dieses Format ist wie SGML für den Datenaustausch prädestiniert, jedoch deutlich einfacher aufgebaut und dementsprechend einfacher zu verarbeiten. Das genaue Format zum Profilaustausch via XML wird durch eine Dokumententyp-Definition (DTD, Document Type Definition) vorgeschrieben, die im Anhang A.3.1 auf Seite 91 zu finden ist.

4.2.2 Anfragevorverarbeitung

Eine wichtige Fähigkeit der Personalisierungskomponente ist es, Nutzeranfragen an das System derart zu verändern, daß sich die Interessen des Nutzers aus dessen Profilen in diesen Anfragen wiederspiegeln. Eine zweite Anforderung besteht darin, Anfragen auszuwerten, wobei häufig gestellt Anfragen so interpretiert werden, daß sich der Nutzer verstärkt für Themen interessiert, nach denen er in der Anfrage sucht, was schließlich zu einer Erweiterung der Interessenprofile führt. Die Aufgabe, Nutzerprofile in Anfragen einzuarbeiten und aufgrund von Anfragen anzupassen, wird durch den Anfragedienst übernommen, dessen Aufbau in Abbildung 4.5 dargestellt ist.

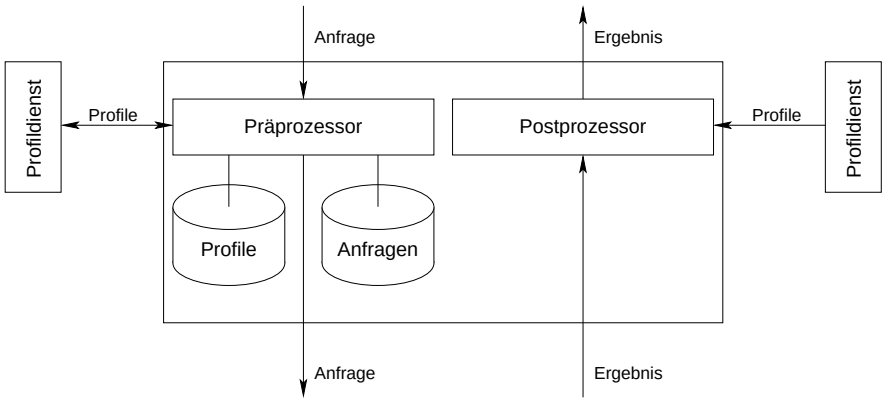

Abbildung 4.5: Aufbau des Anfragedienstes

Der Anfragedienst besteht lediglich aus zwei sogenannten Prozessoren, wobei jeweils einer davon für die Anfragevor- und einer für die Anfragenachbearbeitung benutzt wird. Dabei wird die Anfrage an die Personalisierungskomponente direkt an den Anfragedienst weitergeleitet, von dieser unter Zuhilfenahme des Profildienstes vorverarbeitet und an den Virtuellen Dokumentenserver delegiert. Das Ergebnis der Anfrage erhält die Anfragedienst vom Virtuellen Dokumentenserver zurück, führt eine Nachverarbeitung dieses

Ergebnisses durch und liefert dieses schließlich an das Nutzersystem. An das System gestellte Anfragen werden im Anfragedienst gespeichert. Wird in diesen Anfragen häufig nach bestimmten Begriffen gesucht, so werden diese in das Interessenprofil aufgenommen.

Um eine Anfragevorverarbeitung durchführen zu können, muß die Anfragesprache festgelegt werden. Im folgenden sollen deshalb die Retrieval-Fähigkeiten der Information Retrieval Query Language (IRQL) [HP99] und des Information Retrieval-Anteils von SQL 99 (SQL Multimedia and Application Packages (SQL/MM), Part 2: Full-Text) [sql00] skizziert werden, um danach die Möglichkeiten der Anfragevorverarbeitung konkret vorstellen zu können. Die beiden Anfragesprachen sollen dabei weder bewertet noch miteinander verglichen werden.

4.2.2.1 Information Retrieval Query Language

Die Information Retrieval Query Language (IRQL) wird an der Universität Rostock entwickelt und bereits in Projekten eingesetzt, insbesondere im Projekt BLUEVIEW. Sie koppelt die klassischen SQL-typischen Anfragekonstrukte mit Volltextsuchmöglichkeiten des Information Retrieval.

Diese Anfragesprache ermöglicht die Suche nach Termen (Buchstabe, Wort, Absatz), wobei durch die Angabe einer Grenze festgelegt werden kann, wie häufig die Suchterme im zu durchsuchenden Volltext auftreten dürfen. Die Suchterme können dabei gewichtet werden. Um eine bestimmte Fehlertoleranz bei der Eingabe der Suchterme zuzulassen, kann bei der Suche angegeben werden, ob exakt nach den angegebenen Suchtermen gesucht werden soll, oder auch nach solchen Termen, die davon abweichen. Neben kompletten Termen kann auch nach Teilen (Substrings) davon gesucht werden.

Ein weiterer Schwerpunkt der IRQL besteht in der Möglichkeit der Phonetischen Suche, bei der nach Termen gesucht wird, deren Aussprache genau wie die des Suchterms klingt. Eine Angabe von Ranking-Funktionen, nach denen die Suchergebnisse sortiert werden, wird ebenso unterstützt. Die Möglichkeit der Proximity-Suche besteht außerdem. Dabei werden vom Typ des Volltexts (etwa ein LaTeX-Dokument) abhängige Einheiten, wie ein Satz oder ein Absatz, angegeben, die sich zwischen den Suchtermen befinden sollen.

4.2.2.2 SQL/MM, Part 2: Full-Text

Die hier skizzierte Anfragesprache ist Teil der SQL99 Spezifikation, deren Kern die Volltextsuche auf Basis des Information Retrieval ist. Mit dem Anteil von SQL99 kann unter Verwendung von sprachspezifischen Thesauri nach Termen wie Wörtern oder Absätzen gesucht werden, die

- so klingen, wie der Suchterm (Phonetische Suche)
- allgemeiner oder spezieller sind als der Suchterm
- ein Synonym des Suchterms oder damit verwandt sind
- gegenüber dem Suchterm bevorzugt verwendet werden

- ein Oberthema des Suchtermes darstellen.

Die Suche nach Termen, die relevant zum Suchterm sind (IS-ABOUT-Anfrage) wird ebenfalls unterstützt.

Ein Schwerpunkt des Volltext-Anteils von SQL/MM ist wie bei der IRQL die Proximity-Suche. Auch hier wird nach zwei Mustern (Suchtermen) gesucht, die sich in einem bestimmten Abstand befinden müssen. Die Einheit eines solchen Abstands kann dabei Worte, Sätze oder auch Absätze umfassen. Darüberhinaus wird die Suche mittels Stammwortbildung (Stemming) angeboten, bei der die Worte des Suchterms und des zu durchsuchenden Textes jeweils auf ihre Stammform reduziert werden. Neben dem Stemming kann auch eine Normalisierung durchgeführt werden, etwa um Sonderzeichen zu umgehen, beispielsweise bei der Umformung von *Müller* in *Mueller*.

Ein anderer Bereich von SQL/MM beinhaltet die Suche auf multilingualen Texten. Dabei wird zusätzlich zu jedem Suchterm angegeben, aus welcher Sprache er stammt. Das Ranking ist im Standard nur sehr schwach ausgeprägt, es kann lediglich durch eine Anfrage bestimmt werden, welche Wertigkeit ein Suchterm bezüglich eines zu durchsuchenden Textes aufweist, wobei eine solche Anfrage natürlich auch Teil einer anderen Anfrage sein kann.

4.2.2.3 Einflechtung von Profildaten

Voraussetzung für das Einflechten von Profildaten in Anfragen an das System ist, daß bestimmte Teile des Datenbankentwurfs bekannt sind. Insbesondere müssen Namen und Typen der Attribute bekannt sein, die in die Anfrage eingearbeitet werden sollen. Ausgehend von den vorgestellten Anfragesprachen werden die Profildaten durch Ändern der WHERE-Klausel in Anfragen eingearbeitet. Diese findet dabei in zwei Schritten statt:

1. Einarbeitung der Nutzerinteressen
2. Hinzufügen von Einstellungen aus den Nutzerprofilen

Im ersten Schritt werden die Interessen eines Nutzers, die in seinen Interessenprofilen gespeichert sind, also Metadaten, einer Anfrage konjunktiv hinzugefügt. Die SELECT- und FROM-Klauseln bleiben unverändert. Interessiert sich ein Nutzer beispielsweise für Datenbankliteratur, so könnte die WHERE-Klausel durch die Bedingung

```
AND Subject CONTAINS 'database'
```

ergänzt werden. Der Anfragetyp (*Exact Match*, *Partial Match* etc.) einer so hinzugefügten Bedingung kann dabei beliebig gewählt werden bzw. ist durch den Nutzer anzugeben und wird in den Profileinstellungen gespeichert. Die Verwendung von Exact-Match-Anfragen steigert die Performanz, Partial-Match-Anfragen sind “toleranter” und liefern meist eine größere Ergebnismenge. Nachfolgend werden zwei Anfragen angegeben, wobei die erste Anfrage vom Nutzersystem stammt und die zweite durch Veränderung der ersten entsteht. Mit der ersten Anfrage

```
SELECT ...
FROM   ...
WHERE  Author CONTAINS 'Heuer'
   OR  Author CONTAINS 'Uhrmacher'
```

wird nach Dokumenten gesucht, die von den Autoren HEUER oder UHRMACHER stammen. Interessiert sich der Nutzer nun laut seines Interessenprofils nur für Datenbankenliteratur in deutscher Sprache, die nach 1995 geschrieben wurde, so wird die gegebene Anfrage folgendermaßen erweitert:

```
SELECT ...
FROM   ...
WHERE ( Author CONTAINS 'Heuer'
   OR  Author CONTAINS 'Uhrmacher' )
  AND Subject CONTAINS 'database'
  AND DATE > '1994-12-31'
```

Beim Hinzufügen von neuen Bedingungen anhand von Profilen muß darauf geachtet werden, daß bereits vorhandene Bedingungen nicht negiert werden, da dies zu einem leeren Anfrageergebnis führen würde. Um dies zu vermeiden, muß vor dem Hinzufügen einer Bedingung getestet werden, ob diese schon in negierter Form vorliegt, verglichen mit der neuen Bedingung. Ist das der Fall, dann wird die Bedingung nicht hinzugefügt. Ein mehrmaliges Auftreten einer Bedingung stellt kein Problem dar, da derartige Konstellationen durch den Optimierer des verwendeten Datenbanksystems bereinigt werden.

Nach dem Einarbeiten von Nutzerinteressen werden in einem zweiten Schritt die Einstellungen der Nutzerprofile in die Anfrage eingeflochten, indem die Bedingungen der WHERE-Klausel erweitert werden. Dabei können insbesondere die Retrieval-Fähigkeiten der beiden vorgestellten Anfragesprachen ausgenutzt werden. Beispielsweise können Fehlertoleranzen oder Gewichtungen festgelegt, eine Stammwortreduktion oder eine Wortnormalisierung der Suchterme gefordert werden. Stellt eine Nutzer z. B. in seinem Profil ein, daß in Begriffen, nach denen er sucht, jeweils maximal drei Fehler vorkommen dürfen und daß diese Begriffe mit dem Wert 0.8 gewichtet werden sollen, dann wird die zuletzt angegebene Anfrage unter Benutzung der IRQL nochmals wie folgt erweitert:

```
SELECT ...
FROM   ...
WHERE ( Author CONTAINS 'Heuer' WITH WEIGHT 0.8  3 ERRORS
   OR  Author CONTAINS 'Uhrmacher' WITH WEIGHT 0.8  3 ERRORS )
  AND Subject CONTAINS 'database' WITH WEIGHT 0.8  3 ERRORS
  AND DATE > '1994-12-31'
```

Bei dieser Variante zur Erweiterung der Anfrage mit Nutzereinstellungen werden alle Nutzereinstellungen jeweils allen Bedingungen hinzugefügt. Dies kann jedoch erweitert werden, etwa indem jedem Eintrag im Interessenprofil eine oder mehrere Einstellungen zugeordnet werden, so daß beispielsweise jeder hinzugefügte Suchterm separat gewichtet oder mit einer maximalen Anzahl von akzeptierten Fehlern versehen werden kann.

4.2.2.4 Automatisches Anpassen von Interessenprofilen

Sucht ein Nutzer häufig nach gleichen Begriffen, dann können diese Begriffe als Teil seiner Interessen interpretiert und darum automatisch in seine Interessenprofile aufgenommen werden. Um dies technisch mit Hilfe des Anfragedienstes umzusetzen, werden innerhalb dieses Dienstes alle Nutzeranfragen gespeichert. Bei jeder gestellten Anfrage wird dann überprüft, ob sie oder Teile davon von einem Nutzer[2] bereits mehrmals gestellt

[2] Hierbei werden immer die Anfragen auf einzelne Nutzer bezogen betrachtet.

wurden. Ist dies der Fall, dann wird innerhalb des Anfragedienstes ein Profil des Nutzers erstellt, das die Suchbegriffe enthält, nach denen häufig gesucht wurde. Das Profil wird im Anfragedienst gespeichert und danach an den Profildienst gesendet, wo es wie in Abschnitt 4.2.1 beschrieben, verarbeitet wird.

Verliert ein Nutzer das Interesse an bestimmten Themen, beispielsweise weil er eine Arbeit abgeschlossen hat, dann kann sein Interessenprofil auf zwei verschiedenen Wegen angepaßt werden:

1. Der Nutzer ändert seine Interessenprofile von Hand.

2. Das System erkennt, daß sich der Nutzer für bestimmte Themen nicht mehr interessiert und paßt seine Interessenprofile selbstständig an.

Stellt ein Nutzer nach einer gewissen Zeit keine Anfragen nach Begriffen mehr, nach denen er zuvor häufig gesucht hat und die aufgrund dieser häufigen Suche in seine Interessenprofile eingeflossen sind, dann ist dies als Interessenverlust zu interpretieren. Das System erkennt diesen Verlust dadurch, daß der Nutzer verstärkt nach anderen Begriffen sucht. Deshalb müssen, nach Ablauf einer gewissen Zeit die entsprechenden Interessenprofile des Nutzers und die dazugehörigen Anfragen im Anfragedienst gelöscht werden. Darüberhinaus muß dieser Löschvorgang der Profilverwaltung mitgeteilt werden, damit diese ihre Profile entsprechend anpaßt.

Ändert ein Nutzer seine Interessenprofile dagegen von Hand, dann bleiben die Profile im Anfragedienst zunächst unverändert. Erst wenn sich diese Änderungen in seinem Anfrageverhalten wiederspiegeln, er also nach anderen Themen sucht, dann erkennt der Anfragedienst dies und paßt die Profile dementsprechend an.

4.2.3 Verwaltung lokaler Dokumente

Die Verwaltung lokaler Dokumente wird durch einen Lokalen Dokumentenserver (LDS) übernommen, der in [Tit99] entworfen und hier um verschiedene Fähigkeiten erweitert wird. Nachfolgend sollen zunächst die grundlegenden Eigenschaften eines Lokalen Dokumentenservers erläutert werden, um diesen daraufhin auszubauen und in die Gesamtarchitektur einzufügen.

Der hier erweiterte Lokale Dokumentenserver ist nicht Teil der Personalisierungskomponente, sondern Bestandteil des Virtuellen Dokumentenservers. Der Grund dafür ist, daß die Funktionalität von Lokalen Dokumentenservern eine geeignete Basis darstellt, auf der aufgesetzt werden kann. Außerdem bleibt somit die Anfragebearbeitung samt globalem Ranking alleinige Aufgabe des Virtuellen Dokumentenservers. Die Konzeption einer vollkommen neuen Komponente zur Verwaltung lokaler Dokumente als Bestandteil der Personalisierungskomponente würde u. a. bedeuten, daß die Anfragebearbeitung, insbesondere das Mischen von Anfrageergebnissen verschiedener Quellen einerseits im Virtuellen Dokumentenserver[3] und zusätzlich in der Personalisierungskomponente[4] durchgeführt werden müßte.

[3] Mischen der Ergebnisse verschiedener LDS

[4] Mischen der Ergebnisse von VDS und der Komponente zur Verwaltung lokaler Dokumente

4.2.3.1 Lokaler Dokumentenserver

Ein Lokaler Dokumentenserver speichert nach [Tit99] Dokumente und dazugehörige Metadaten. Zum Zugriff darauf werden über eine Schnittstelle Methoden bereitgestellt, um exakt oder auch mit Methoden des Information Retrieval über den Metadaten zu suchen und auf Dokumente zuzugreifen. Der Zugriff auf Dokumente erfolgt dabei über einen Dokumentenschlüssel, der als Ergebnis einer Metadatenanfrage zurückgeliefert wird.

Wie in Abbildung 4.6 dargestellt, werden mehrere Lokale Dokumentenserver (LDS) einem Virtuellen Dokumentenserver (VDS) zugeordnet. Metadatenanfragen an den Virtuellen Dokumentenserver werden auf die verschiedenen LDS verteilt, Anfrageergebnisse werden anschließend vom Virtuellen Dokumentenserver homogenisiert, gemischt und schließlich zurückgegeben. Dokumenten-Retrieval-Anfragen werden dagegen nicht direkt auf beteiligte LDS verteilt, sondern an den Replikations- und Cachingdienst gestellt. Zusätzlich zur Anfrage nach dem Dokument wird die Kennung des Nutzers übergeben, der diese Anfrage gestellt hat. Der Replikations- und Cachingdienst holt anhand des Dokumentenschlüssels das geforderte Dokument und die dazugehörigen Zugriffsrechte entweder aus einem der Lokalen Dokumentenserver oder von einem externen VDS, überprüft die Zugriffsrechte des Nutzers und gibt im positiven Fall das Dokument zurück.

Repliziert der Replikationsdienst Dokumente von externen Virtuellen Dokumentenservern, so kann er diese nicht nur in einem Zwischenspeicher (Cache) ablegen, sondern auch dauerhaft auf einem speziell dafür vorgesehenen Lokalen Dokumentenserver (LDS_R) gehalten. Dadurch wird die Zugriffszeit auf das Dokument in Zukunft verkürzt, die Performanz somit gesteigert und eine Autonomie gegenüber dem entfernten VDS erreicht, so daß beispielsweise trotz eines Ausfalls des entfernten VDS das Dokument verfügbar ist.

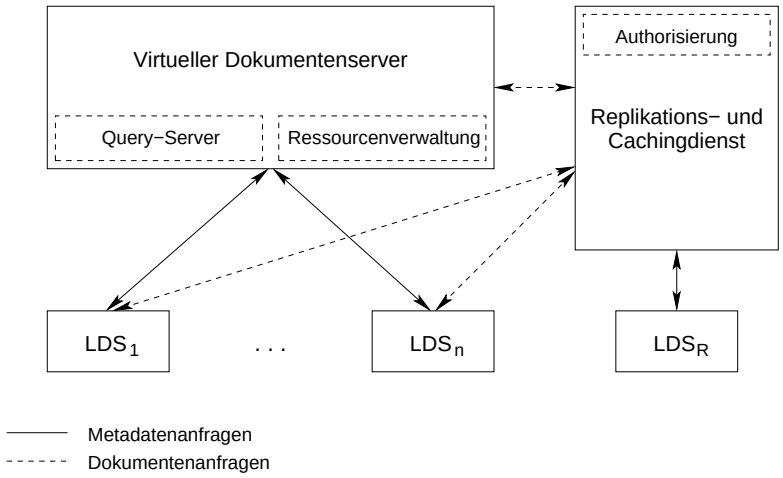

Abbildung 4.6: Einordnung der Lokalen Dokumentenserver

4.2.3.2 Erweiterung des LDS

Um lokale Dokumente nutzergesteuert mit Hilfe eines Lokalen Dokumentenservers so verwalten zu können, wie in Abschnitt 3.3.3 beschrieben, muß dieser erweitert werden. Diese Erweiterungen betreffen neben der Fähigkeit, vorhandene Dokumente und Metadaten zu verwalten und Anfragen auf ihnen auszuführen auch die folgenden Möglichkeiten:

- Einfügen, Ändern und Löschen von Dokumenten und dazugehörigen Metadaten unter Vermeidung von Duplikaten.
- Benachrichtigung von Nutzern über Änderungen am Dokumentenbestand des LDS
- Kontrolle des Zugriffs auf Dokumente durch Vergabe von Rechten

Unter dem Ändern eines Dokumentes soll hier das Einfügen einer neuen Version verstanden werden, wobei die alte Version erhalten bleibt. Um dies zu realisieren, bedient sich der erweiterte LDS eines Versionierungsmechanismus, d. h. jedes eingefügte Dokument entspricht einer Version und auf jede dieser Versionen kann zugegriffen werden. Nähere Erläuterungen zur Versionsverwaltung sind im nächsten Abschnitt zu finden.

Abbildung 4.7 gibt einen Überblick über die Erweiterungen des LDS, die in den folgenden Abschnitten beschrieben werden und zeigt, an welcher Stelle der erweiterte LDS (LDS_E) im System eingeordnet wird. Der Aufbau des erweiterten LDS ist in Ab-

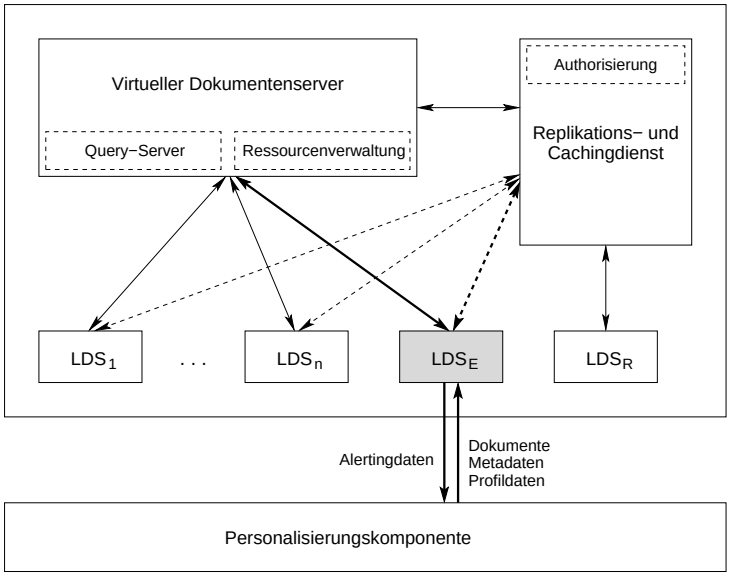

Abbildung 4.7: Einordnung des erweiterten LDS

bildung 4.8 dargestellt. Er besteht zum einen aus einem Dokumentenmanager, der Dokumente und dazugehörige Metadaten verwaltet, und zum anderen aus einem Profilmanager, der die Nutzerprofile verwaltet. Während der Profilmanager über Profiländerungen vom Profildienst informiert wird, ist es die Aufgabe des Dokumentenmanagers, Anfragen des Virtuellen Dokumentenservers zu beantworten, Änderungen am Dokumentenbe-

stand zu bearbeiten, die über das Nutzersystem durchgeführt werden und die Alerting-Komponente über diese Änderungen gemäß der Profildaten des Profilmanagers zu informieren.

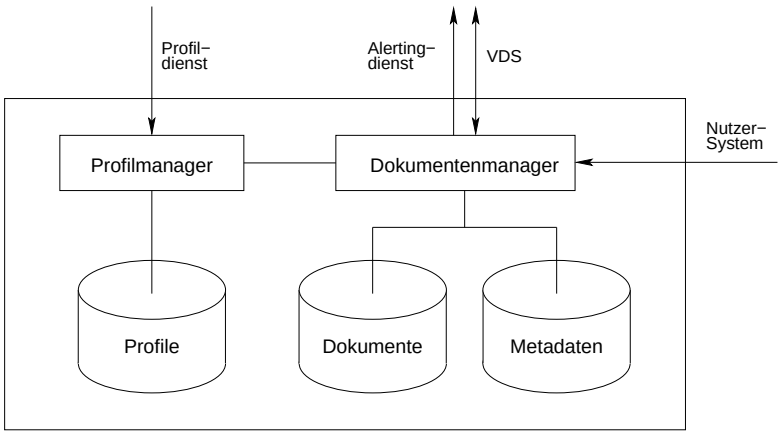

Abbildung 4.8: Aufbau des erweiterten LDS

4.2.3.3 Einfügen, Ändern und Löschen von Dokumenten

Dokument und Metadaten werden immer getrennt voneinander gespeichert. Über einen Dokumentenschlüssel, der Dokumenten-ID, wird festgelegt, welche Metadaten zu welchem Dokument gehören und umgekehrt. Außerdem wird jedes Dokument eines LDS über eine solche eindeutige Dokumenten-ID identifiziert. Dabei wird zwischen der ID unterschieden, mit der ein Dokument mit all seinen Versionen identifiziert wird und der ID, die eine einzelne Version eines Dokumentes auszeichnet. Somit sind mehrere Versions-IDs immer genau einer Dokumenten-ID zugeordnet und umgekehrt. Ein Dokument ist damit also eine logische Einheit, der mehrere physische zugehörig sind.

Beim Einfügen eines neuen Dokumentes werden neben dem Dokument und seinen Metadaten der Besitzer des Dokumentes, eine Versionsnummer und die Nutzer angegeben, die neben dem Besitzer des Dokumentes Zugriff darauf[5] erhalten sollen. Als Besitzer wird hier der Nutzer betrachtet, der das Dokument in den Bestand einfügt. Zurückgegeben wird eine ID, die das Dokument mit all seinen Versionen identifiziert und für den Zugriff auf dieses Dokument notwendig ist. Sie wird innerhalb der Personalisierungskomponente gespeichert und zum Zugriff darauf benötigt, etwa um das Dokument zu ändern oder aus dem Bestand zu entfernen. Intern wird für auch das physische Dokument eine ID vergeben, mit der die Version identifiziert wird.

Um zu vermeiden, daß physische Dokumente mehrfach gespeichert werden, wird für das einzufügende Dokument eine Prüfsumme errechnet und zusammen mit dem Dokument abgespeichert. Da solch eine Prüfsumme eindeutig für ein physisches Dokument ist, kann beim Einfügen eines neuen Dokumentes anhand der Prüfsummen getestet werden, ob sich das Dokument bereits im Bestand befindet (siehe dazu Abschnitt 4.2.3.4).

[5] Die Zugriffsrechte werden immer für einzelne Versionen eines Dokumentes vergeben.

Augrund der Eindeutigkeit dieser Prüfsumme kann sie außerdem als interne ID benutzt werden. Nach dem Einfügen des Dokumentes wird es indexiert, d. h. es wird ein Index für das Dokument aufgebaut und in den Gesamtindex aller Dokumente eingefügt. Damit wird eine Volltextsuche möglich.

Das Ändern eines bereits im System gespeicherten Dokumentes besteht aus dem Einfügen einer neuen Version. Dabei wird das neue Dokument zusammen mit einer Versionsnummer in das System eingefügt. Voraussetzung dafür ist jedoch, daß der Nutzer, der die neue Version hinzufügen möchte, Besitzer des Dokumentes ist. Abschließend die eingefügte Dokumentenversion indexiert, um die Volltextsuche über den einzelnen Versionen eines logischen Dokumentes zu ermöglichen.

Beim Löschen eines Dokumentes werden alle Versionen dieses Dokumentes inklusive der dazugehörigen Metadaten entfernt. Da nur der Besitzer des Dokumentes dieses auch löschen kann, muß die Kennung des Nutzers überprüft werden, der das Dokument löschen möchte. Wie beim Update, so wird auch das Löschen eines Dokumentes nur dann durchgeführt, wenn der Nutzer identisch mit dem Besitzer des Dokumentes ist. Die Volltextindizes, die für die einzelnen Versionen des Dokumentes angelegt und dem Gesamtindex zugefügt wurden, werden aus diesem Gesamtindex entfernt. Sollte das verwendete System zur Verwaltung von Dokumenten (Datenbank- oder Volltextsystem) nicht in der Lage sein, Teile aus seinem Index wieder zu löschen, so muß dieser Index komplett gelöscht und wieder neu aufgebaut werden.

Das Anpassen des Volltextindex bzw. das Entfernen und Neuaufbauen geschieht dabei jedoch nicht unmittelbar nach Löschen des Dokumentes, sondern verzögert. Auf diese Weise kann eine Steigerung der Systemrobustheit und der Performanz des Systems erreicht werden. Da das Löschen von Teilen aus dem Index bzw. der Neuaufbau eines Index sehr aufwendig ist, bringt ein verzögertes Ändern bzw. ein Verzögern des Neuaufbaus einen Zeitvorteil, da Änderungen nicht einzeln sondern komplett bearbeitet werden.

Zusätzlich zum Einfügen, Ändern und Löschen eines Dokumentes können die Menge der zugriffsberechtigten Nutzer geändert und Metadaten in den lokalen Bestand eingefügt werden, die durch das Alerting an die Personalisierungskomponente übertragen wurden. Diese Metadaten müssen jedoch entweder eine Dokumenten-ID enthalten, aus der hervorgeht, auf welchem externen Dokumentenserver sich das dazugehörige Dokument befindet oder einen Hinweis darauf, daß dieses Dokument nicht verfügbar ist, beispielsweise weil es nicht in elektronischer Form vorliegt.

4.2.3.4 Vermeidung von Duplikaten

Da mehrere Nutzer dasselbe Dokument in das Bibliothekssystem einfügen können, kann das System Duplikate enthalten. Um dies zu vermeiden, muß vor dem Einfügen eines Dokumentes sichergestellt werden, daß das zu speichernde Dokument noch nicht im System vorhanden ist. Dies könnte beispielsweise überprüft werden, indem die Metadaten des neuen Dokumentes mit den schon vorhandenen Metadaten verglichen werden. Da aber verschiedene Nutzer u. U. unterschiedliche Metadaten für dasselbe Dokument angeben, scheidet diese Variante aus. Eine andere Möglichkeit, Duplikate zu erkennen, besteht nun darin, eine Prüfsumme des einzufügenden Dokumentes zu berechnen und diese mit den Prüfsummen der bereits im System gespeicherten Dokumente zu vergleichen. Dies ließe sich etwa mittels des MD5-Verfahrens [Riv92] realisieren, daß zu einem Dokument bzw.

zu einer Zeichenkette eine eindeutige ID berechnet, die vom Inhalt des Dokumentes bzw. der Zeichenkette abhängt.

Stellt sich beim Einfügen eines physischen Dokumentes heraus, daß dieses bereits vorhanden ist, so wird es nicht ein zweites Mal eingefügt. Beim Löschen des Dokumentes durch einen Nutzer muß überprüft werden, ob es nicht mehr referenziert wird, denn nur dann kann es entfernt werden. Wird das Dokument noch referenziert, d. h. haben mehrere Nutzer ein und dasselbe Dokument eingefügt,[6] dann bleibt es im Bestand vorhanden und es wird nur der Verweis darauf entfernt, der beim Einfügen angelegt wurde.

4.2.3.5 Benachrichtigung

Um Nutzer darüber informieren zu können, daß sich Änderungen am lokalen Datenbestand ergeben haben, die für sie von Interesse sind, müssen die Interessen des Nutzers zunächst bestimmt und, wie in Abschnitt 4.2.1 beschrieben, in Profilen abgelegt werden. Diese Profile werden dann generalisiert, an einen entfernten Alerting-Server (beispielsweise Hermes, siehe Abschnitt 4.2.4.1) und an den hier beschriebenen erweiterten Lokalen Dokumentenserver gesendet. Dieser speichert die Profile ab und überprüft bei jeder Änderung[7] des Datenbestandes, ob Profile vorhanden sind, auf die diese Änderungen passen[8]. Ist dies der Fall, so werden die Dokumentenänderungen und der Name des Profils, auf das die Änderung paßt, an die Alerting-Komponente gesandt. Zusätzlich dazu werden die Kennungen der Nutzer übermittelt, die Zugriff auf das Dokument haben bzw. die Kennungen der Nutzer, denen durch die Änderung das Zugriffsrecht darauf entzogen wurde. Die Alerting-Komponente wertet diese Nachricht aus und informiert letztlich die Nutzer.

Ändert ein Nutzer sein Profil und ergeben sich infolge dessen Änderungen am generalisierten Profil, so werden dem erweiterten LDS die neuen Profildaten mitgeteilt. Dieser ersetzt daraufhin alle Daten des betroffenen Profils durch die neuen Daten. Wird ein generalisiertes Profil im Profildienst gelöscht, so muß dieses im Benachrichtigungsdienst, dem erweiterten LDS und dem externen Hermes-Server gelöscht werden.

4.2.3.6 Anbindung des erweiterten LDS

Nach der Implementierung des hier beschriebenen erweiterten Lokalen Dokumentenservers muß dieser im Virtuellen Dokumentenserver des vorhandenen Systems eingebunden werden. Der VDS stellt dazu eine Möglichkeit bereit, mit der beliebige LDS zur Laufzeit des Systems angemeldet werden können. Bei der Anmeldung prüft die Ressourcenverwaltung, ob sie eine Verbindung zum neuen LDS herstellen und auf seine Daten zugreifen kann.[9] Gelingt dies, dann vergibt die Ressourcenverwaltung eine ID für den neuen LDS, meldet diesen beim Anfrageserver des VDS an und liefert die vergebene ID zurück. Durch die Anmeldung beim Anfrageserver wird es ermöglicht, daß Anfragen an den VDS zukünftig auch an den neuen LDS weitergeleitet werden. Mit der Vergabe einer ID steht u. a. die Möglichkeit offen, daß beim Stellen einer Anfrage an den VDS festgelegt werden

[6] Versucht ein Nutzer, ein Dokument einzufügen, dann führt dies für ihn immer zum Erfolg, auch wenn es bereits vorhanden ist und deshalb physisch nicht ein weiteres Mal gespeichert wird.

[7] Insert, Update, Delete von Dokumenten und Ändern der Zugriffsrechte auf Dokumente

[8] Geeignete Verfahren dazu werden in Anhang A.2 diskutiert.

[9] Zwischen VDS und Erweitertem LDS besteht eine typische Client-Server-Beziehung, wobei der VDS die Rolle des Client, der LDS die des Servers übernimmt.

kann, von welchen Ressourcen, also von welchen LDS diese Anfrage bearbeitet werden soll. Genaueres dazu findet sich im folgenden Abschnitt bzw. in [Tit99].

4.2.3.7 Anfragebearbeitung

Der Lokale Dokumentenserver ist durch seine Konzeption in [Tit99] bereits vollständig in der Lage, Anfragen zu bearbeiten. Dabei werden insbesondere auch die Zugriffsrechte der Dokumente beachtet, so daß ausschließlich berechtigte Nutzer Zugriff darauf erhalten.

Um zu erreichen, daß, wie in Abschnitt 3.3.3 gefordert, explizit auf dem erweiterten LDS gesucht werden kann, wird beim Übergeben der Anfrage an den Virtuellen Dokumentenserver in der Liste der Ressourcen, auf denen die Anfrage auszuführen ist, ausschließlich die ID des erweiterten LDS angegeben. Die Vergabe dieser ID erfolgt, wie oben beschrieben, bei der Anmeldung des LDS in der Ressourcenverwaltung des Virtuellen Dokumentenservers.

4.2.4 Benachrichtigung

Die Aufgabe der Benachrichtigung wird von der Benachrichtigungskomponente bzw. durch den von ihr bereitgestellten Benachrichtigungsdienst übernommen, der Teil der hier vorgestellten Personalisierungskomponente ist. Die Benachrichtigung bzw. das Alerting kann dabei in das lokale und das globale Alerting unterteilt werden. Während beim globalen Alerting der Nutzer darüber informiert wird, daß sich an externen Dokumentenbeständen Änderungen ergeben haben, die das Hinzufügen neuer Dokumente und deren Metadaten beinhalten, wird er beim lokalen Alerting über Änderungen am lokalen Datenbestand informiert. Dabei können Änderungen am lokalen Datenbestand neben dem Hinzufügen von Dokumenten und dazugehörigen Metadaten auch das Ändern und Entfernen dieser Ressourcen sowie die Änderung von Zugriffsrechten darauf bedeuten.

Der komplette Vorgang des Alertings besteht nun aus den folgenden Schritten:

- Konfiguration des Benachrichtigungsdienstes
- Lokales bzw. globales Alerting
- Auslieferung der Benachrichtigungsdaten an die Nutzer

Bei der Konfiguration des Benachrichtigungsdienstes legt der Nutzer fest, worüber er informiert werden möchte. Diese Informationen werden im Interessenprofil des Nutzers innerhalb des Profildienstes abgelegt und in komprimierter Form[10] an den Benachrichtigungsdienst weitergeleitet. Ergeben sich Änderungen an Dokumentenbeständen, so wird der hier vorgestellte Alerting-Dienst darüber informiert. Kommen diese Informationen von einem externen Dienst, dann wird diese Art der Benachrichtigung als *globales Alerting* bezeichnet, bei Daten vom erweiterten Lokalen Dokumentenserver entsprechend als *lokales Alerting*. Nach einer Auswertung dieser Daten durch einen Vergleich mit den Nutzerprofilen werden sie schließlich an die Nutzer ausgeliefert, die sich laut ihrer Profile dafür interessieren.

[10] Der Profildienst faßt mehrere ähnliche Profile durch Generalisierung zusammen (siehe Abschnitt 4.2.1.3).

Zur Realisierung der globalen Benachrichtigung wird der von der Freien Universität Berlin bereitgestellte Alerting-Dienst *HERMES* eingebunden, dessen Aufbau und Funktionsweise im nächsten Teilabschnitt skizziert werden. Der Aufbau eines lokalen Alerting-Dienstes ist dagegen eine Neuentwicklung als Bestandteil dieser Arbeit, ebenso die Zusammenfassung von lokalem und globalem Benachrichtigungsdienst in einer Alerting-Komponente, deren Aufbau in Abbildung 4.9 dargestellt ist. Diese Komponente, die den Alerting- bzw. Benachrichtigungs-Dienst bereitstellt, besteht im Kern aus einer Alerting-Datenbank, in der die Daten verzeichnet sind, die entweder durch den Notifier oder bei der Systemanmeldung über den Login-Handler an die Nutzer ausgeliefert werden sollen. Vom Hermes-Server oder dem erweiterten Lokalen Dokumentenserver gelieferte Alerting-Daten werden vom Alerting-Handler entgegengenommen und ausgewertet. Die dazu notwendigen Profildaten werden über den Profile-Handler vom Profildienst geholt. Metadaten, die durch das globale Alerting gewonnen werden, werden durch den Alerting-Handler an den erweiterten LDS übergeben, der diese in den lokalen Metadatenbestand einfügt.

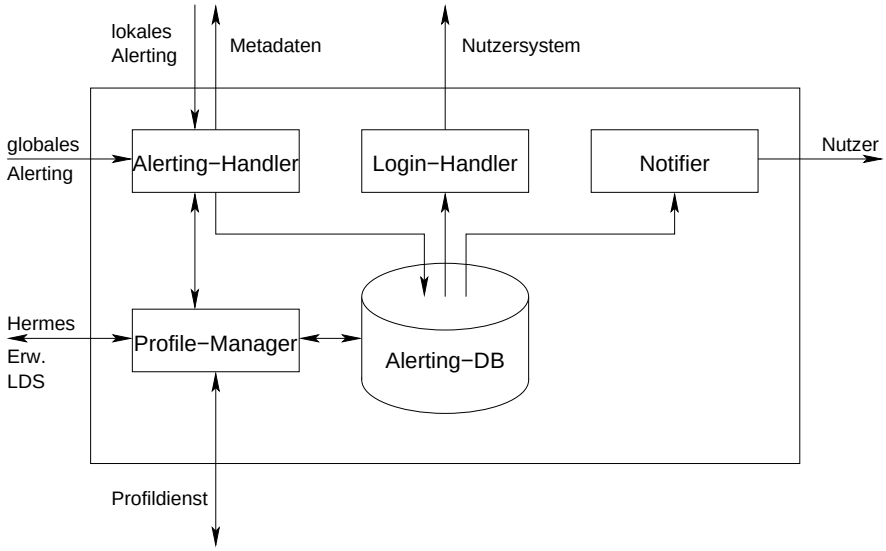

Abbildung 4.9: Aufbau der Alerting-Komponente

4.2.4.1 Das HERMES-Projekt

Das Projekt HERMES[11] [FFS+01] ist ein Teilprojekt des GlobalInfo-Programmes, das an der Freien Universität Berlin durchgeführt wird, und dessen Ziel der Aufbau eines Benachrichtigungsdienstes ist. Dieser Dienst soll in der Lage sein, beliebige Nutzer selbstständig zu informieren, wenn neue Dokumente in bestehende Dokumentenbestände eingefügt werden.

Der Alerting-Service ist momentan an verschiedene Informationsanbieter gekoppelt, unter denen sich beispielsweise der *Springer-Verlag*, die *ACM* oder auch *Nature* befin-

[11] *http://hermes.inf-fu.berlin.de*

den. Fügen diese Anbieter ihren Beständen neue Dokumente hinzu, dann wird dies vom Hermes-Server registriert. Grundsätzlich wird dabei zwischen folgenden Typen von Informationsanbietern unterschieden:

Kooperativ Die Daten, mit denen die Änderungen an Dokumentenbeständen beschrieben werden, folgen einem vorgegebenem Format.

Nicht-kooperativ Das Format der Änderungsbeschreibungen folgt keinem zentral vorgegebenen, sondern wird durch den Informationsanbieter festgelegt.

Aktiv Aktive Informationsanbieter informieren den Hermes-Server selbständig, wenn Änderungen an ihren Beständen vorgenommen wurden.

Passiv Passive Informationsanbieter stellen die Informationen zu vorgenommenen Änderungen an ihren Datenbeständen bereit, informieren den Hermes-Server jedoch nicht darüber, so daß dieser in bestimmten Zeitabständen nach derartigen Informationen Ausschau halten muß.

Informationsanbieter sind immer aktiv oder passiv und gleichzeitig kooperativ oder nichtkooperativ.

Unabhängig davon, wie und in welchem Format der Hermes-Server über Änderungen an Datenbeständen informiert wurde, überprüft er anhand von Nutzerprofilen, an welche Nutzer er diese Informationen in welcher Weise weiterleiten soll. Dem Nutzer stehen bei der Konfiguration des Dienstes über diese Profile verschiedene Möglichkeiten offen. So kann er das Format (HTML, XML, Plain-Text) und das Medium (momentan ausschließlich E-Mail) wählen, über das er benachrichtigt werden möchte. Außerdem kann er festlegen, wie oft er über Änderungen informiert werden will (täglich, wöchentlich, monatlich). Kern der Profile ist jedoch die Festlegung von Interessen, bei denen der Nutzer mit logisch verknüpften Begriffen beschreiben kann, über welche Änderungen er Informationen wünscht.

4.2.4.2 Konfiguration des Alerting-Dienstes

Die Konfiguration des Alerting-Dienstes erfolgt über den Profildienst, der bereits in Abschnitt 4.2.1 beschrieben wurde. Der Nutzer legt dabei fest, für welche Dokumente bzw. welche Dokumentenbestände er sich interessiert, in welcher Weise (E-Mail, Hinweis bei Systemanmeldung) und in welchen Zeitabständen (täglich, wöchentlich etc.) er über Änderungen dieser Bestände informiert werden möchte (siehe Abschnitt 3.3.4 ab Seite 11). Möchte ein Nutzer beispielsweise wöchentlich per E-Mail informiert werden, wenn sich Änderungen an Dokumenten zu den Gebieten *Datenbanken* und *Digitale Bibliotheken* ergeben, dann könnten seine Konfigurationsdaten für den Alerting-Dienst folgendermaßen aussehen:

Interessen	Auslieferung	Format
Datenbanken Digitale Bibliotheken	wöchentlich	E-Mail

Der Profildienst speichert die Interessen der Nutzer zunächst in sogenannten Nutzerprofilen ab. Im Anschluß daran sucht er nach Profilen, die sich ähnlich sind, wobei unter Ähnlichkeit das in Abschnitt 4.2.1.2 eingeführte Maß zu verstehen ist. Nach dem in Abschnitt 4.2.1.3 vorgestellten Algorithmus werden dann ähnliche Profile zu einem einzigen Profil generalisiert, welches abgespeichert und an den Profile-Manager des Benachrichtigungsdienstes übergeben wird. Dieser legt das generalisierte Profil sowohl auf dem Hermes-Server als auch im erweiterten Lokalen Dokumentenserver aus Abschnitt 4.2.3 ab.

Auf dem Hermes-Server werden ausschließlich generalisierte Profile abgelegt, um eine Verteilung der Last zu erreichen, da so die Menge der Profile auf dem Hermes-Server reduziert wird. Ein Großteil der Last für das globale Alerting wird somit auf die hier vorgestellte Alerting-Komponente verlagert, wobei sich diese Verlagerung jedoch nur auf lokale Nutzer bezieht, nicht auf alle Benutzer des Hermes-Service. Die Benutzung von generalisierten Profilen für das lokale Alerting zielt ebenfalls auf Lastverteilung ab, da so der Großteil des Alerting-Vorgangs von der Alerting-Komponente bearbeitet wird, und nicht vom Profildienst oder dem erweiterten LDS.

Ändert ein Nutzer seine Interessen und damit sein Profil, dann löscht die Profilverwaltung das damit verbundene allgemeine Profil und generalisiert die Profile neu, die in das allgemeine Profil eingeflossen sind. Daraufhin übermittelt sie die Profildaten des neu generalisierten Profils an den Benachrichtigungsdienst. Dieser speichert die neuen Daten ab und sendet sie an den Hermes-Server und den erweiterten Lokalen Dokumentenserver. Der folgende Algorithmus faßt in einer Pseudonotation die Schritte zusammen, die beim Ändern eines Profils durchzuführen sind:

Algorithmus 4.3 (Änderung von Profilen)

```
changeProfile(profileID old, profileID new)
    profileID  genProfile = old.getGeneralizedProfile
    old.delete
    profileSet profiles = genProfile.getConcreteProfiles

    genProfile.deleteOn(HERMES)
    genProfile.deleteOn(ExtendedLDS)
    genProfile.delete

    profiles.add(new)
    profileSet newGeneraliezd = profiles.generalize
    foreach p in newGeneralized do
        p.save
        p.sendTo(HERMES)
        p.sendTo(ExtendedLDS)
    od
```

Nach jeder Änderung eines Profils kann es vorkommen, daß sich in der Alerting-Datenbank ein auszuliefernder Eintrag befindet, der nicht mehr auf das neue Nutzerprofil paßt. Aus diesem Grund muß die Alerting-Komponente bei jeder Profiländerung überprüfen, ob vorhandene Einträge der Alerting-DB und dazugehörige Nutzerprofile zueinander passen. Ist dies nicht der Fall, so müssen alle Einträge gelöscht werden, die nicht mehr mit dem Profil übereinstimmen.

Löscht ein Nutzer sein Profil, so löscht die Profilverwaltung das dazugehörige allgemeine Profil. Alle Profile, die an der Generalisierung dieses allgemeinen Profils beteiligt waren, werden anschließend neu generalisiert. Daraufhin wird der Benachrichtigungsdienst darüber informiert, daß das allgemeine Profil zu entfernen ist. Dieser leitet diese Nachricht letztlich an den Hermes-Server und den erweiterten LDS weiter. Allgemeine Profile, die durch die erneute Generalisierung entstehen, werden außer im Profildienst auch im Benachrichtigungsdienst abgelegt. Abschließend müssen aus der Alerting-Datenbank die Einträge des Nutzers entfernt werden, der sein Profil gelöscht hat, um zu verhindern, daß Nutzer über Änderungen an Dokumentenbeständen informiert werden, für die sie sich nicht mehr interessieren.

4.2.4.3 Globales Alerting

Bei der Benachrichtigung über Änderungen an externen Datenbeständen übermittelt der Hermes-Server Daten an den hier vorgestellten Benachrichtigungsdienst, die Informationen über diese Änderungen enthalten. Diese Daten werden mit den konkreten Profildaten aus dem Profildienst verglichen, um zu entscheiden, welcher Nutzer informiert werden soll. Für jeden zu informierenden Nutzer wird ein Eintrag in die Alerting-Datenbank geschrieben, der neben der Nutzerkennung auch die Informationen enthält, die an den Nutzer weitergegeben werden. Außerdem enthält ein solcher Eintrag auch Informationen darüber, aufgrund welchen allgemeinen Profils der Eintrag zustande kam, wann der Nutzer informiert werden soll und in welcher Weise, also per E-Mail oder bei der Systemanmeldung.

Der letzte Schritt des globalen Alertings besteht im Update des lokalen Metadatenbestandes. Dabei werden die Metadaten, die innerhalb der Alerting-Daten des Hermes-Servers übermittelt wurden, dem lokalen Metadatenbestand zugefügt. Das Ziel dieses Updates ist es, nach einer “gewissen” Zeit einen Bestand an Metadaten aufzubauen, der nicht nur die lokal gespeicherten Dokumente beschreibt, sondern auch die Interessen der Nutzer bezüglich extern vorliegender Dokumente wiederspiegelt. Die Suche nach Metadaten kann damit, wie in Abschnitt 3.3.4 beschrieben, zu großen Teilen auf den lokalen Metadaten ausgeführt werden.

Ein Sonderbehandlung ist bei der Benachrichtigung von Nutzern notwendig, die ihre Profile nicht im Bibliothekensystem sondern auf ihrem lokalen Rechner speichern. In diesem Fall können die Alerting-Daten nicht zu einem vom System bestimmten Zeitpunkt gegen die Interessenprofile der entsprechenden Nutzer geparst werden. Deshalb werden Alerting-Daten, die auf generalisierte Profile passen, aber nicht unmittelbar gegen die dazugehörigen konkreten Profile geparst werden können, im System gespeichert. Erst dann, wenn sich der Nutzer beim System anmeldet und infolge dessen sein Profil vom System importiert und an den Benachrichtigungsdienst übermittelt wird, können die Alerting-Daten gegen diese Profile geparst werden. Verläuft dieses Parsing positiv, d. h. entsprechen sie den im Profil angegebenen Interessen des Nutzers, dann werden wie oben beschrieben, die auszuliefernden Daten in die Alerting-Datenbank geschrieben.

4.2.4.4 Lokales Alerting

Beim lokalen Alerting stammen die Daten aus dem erweiterten Lokalen Dokumentenserver, dessen Aufgabe die Verwaltung von lokalen Dokumenten ist (Abschnitt 4.2.3). Darin sind neben den Metadaten des Dokumentes, das geändert wurde, auch Daten enthalten, aus denen hervorgeht, welche Nutzer Zugriffsrechte für dieses Dokument besitzen und in welcher Weise das Dokument geändert wurde. Dies kann neben dem Ändern oder Löschen des Dokumentes auch die Änderung von Zugriffsrechten darauf sein.

Um zu erreichen, daß nur Nutzer über Änderungen an lokalen Dokumenten informiert werden, die auch zum Zugriff darauf berechtigt sind, müssen in den Alerting-Daten des erweiterten LDS auch die Kennungen der Nutzer enthalten sein, denen der Zugriff auf die entsprechenden Dokumente gewährt wird. Daneben werden auch die Kennungen der Nutzer mitgeliefert, denen das Zugriffsrecht entzogen wurde. Diese Nutzer werden dann lediglich darüber informiert, daß sie nicht mehr auf das entsprechende Dokument zugreifen dürfen.

Der eigentliche Vorgang des lokalen Alertings besteht wie beim globalen Alerting darin, die Alerting-Daten mit den Nutzerprofilen zu vergleichen, um herauszufinden, welche Nutzer über die Änderungen informiert werden sollen. Im Unterschied zum globalen Alerting muß hier jedoch, wie oben beschrieben, zusätzlich auf die Zugriffsberechtigungen geachtet werden.[12] Im Anschluß daran wird für jeden zu informierenden Nutzer ein Eintrag in die Alerting-Datenbank eingefügt. Diese Einträge enthalten zusätzlich zu denen des globalen Alertings Informationen über die Art der Dokumentenänderung. Beim Einfügen eines solchen Eintrags ist darauf zu achten, daß es in der Alerting-Datenbank nicht zu Widersprüchen kommt. Ein Widerspruch besteht dann, wenn zwei Einträge für dasselbe Dokument vorkommen, wobei einer durch das Einfügen und der andere das Löschen dieses Dokumentes zustande gekommen ist. Führt das Einfügen eines Eintrags zu einem Widerspruch, so werden beide Einträge entfernt. Wird also beispielsweise ein Dokument eingefügt und kurze Zeit später wieder gelöscht, so werden die Nutzer darüber nicht informiert.

Eine weitere zu vermeidende Konstellation ensteht dann, wenn sich zwei Einträge in der Alerting-Datenbank befinden, wobei einer durch das Update eines Dokumentes und der zweite durch das Löschen desselben Dokumentes entstanden ist. In diesem Falls ist der Eintrag, der aufgrund des Dokumenten-Updates entstanden ist, zu entfernen. Wird ein Dokument also auf einen neuen Stand gebracht (Update) und im Anschluß daran gelöscht, so werden Nutzer, die an diesem Dokument interessiert sind, nur über den Löschvorgang in Kenntnis gesetzt.

Beim lokalen Alerting kann ebenso wie beim globalen Alerting die Situation auftreten, daß Profile, gegen die die Alerting-Daten geparst werden sollen, nicht im System vorhanden sind, da sie lokal beim Nutzer gespeichert werden. Das Vorgehen in dieser Situation ist jedoch identisch zu dem beim globalen Alerting, d. h. die Alerting-Daten werden im System gespeichert und erst dann gegen die “fehlenden” Profile geparst, wenn sie vorhanden sind, also der Nutzer beim System angemeldet ist.

[12] Da die Metadaten von Dokumenten großer Informationsanbieter wie Verlagen i. allg. frei sind (im Gegensatz zu den Dokumenten), müssen beim globalen Alerting keine Zugriffsrechte überprüft werden.

4.2.4.5 Grundlegender Algorithmus

Nachfolgend ist der Algorithmus zu finden, nach dem das Alerting grundsätzlich abläuft. Er unterscheidet sich bei lokalem und globalem Alerting im Wesentlich nur darin, daß der Grund des lokalen Alertings auch das Ändern oder Löschen eines Dokumentes sein kann, während der Grund des globalen Alertings ausschließlich das Hinzufügen eines Dokumentes zu einem Dokumentenbestand ist. Beim lokalen Alerting muß also zusätzlich überprüft werden, um welche Art der Änderung es sich handelt.

Algorithmus 4.4 (Alerting)

```
alert(AlertingData data)
    ProfileID  genProfile = data.getMatchingProfile
    ProfileSet profiles = genProfile.getConcreteProfiles

    foreach p in profiles do
        if p.matches(data) then
            writeAlertingEntry(p.getUser, data.getContent)
        fi
    od
```

4.2.4.6 Auslieferung der Daten

Die Aufgabe des Auslieferns von Alerting-Informationen an die Nutzer übernimmt der sogenannte *Notifier*. Dieser beginnt zu bestimmten Zeiten (etwa um Mitternacht) damit, Einträge aus der Alerting-Datenbank auszulesen und den Inhalt der Einträge an die Nutzer zu senden, die in diesem Eintrag spezifiziert sind. Im Anschluß daran werden die Einträge aus der Datenbank gelöscht. Unter dem Senden von Informationen an den Nutzer soll hier das verschicken von E-Mails verstanden werden. Denkbar ist beispielsweise auch das Versenden einer Nachricht via SMS (Short Message Service) oder Ähnliches.

Eine weitere Variante der Auslieferung von Informationen über Änderungen von Dokumentenbeständen beinhaltet das Informieren von Nutzern über das Nutzersystem. Meldet sich ein Nutzer am Bibliothekssystem an, so wird er durch das Nutzersystem über die Änderungen der Datenbestände informiert, für die er sich subskribiert hat. Die entsprechenden Einträge in der Alerting-Datenbank werden daraufhin gelöscht.

Speichert der Nutzer, der sich am System anmeldet, seine Profile (oder zumindest einige seiner Profile) außerhalb des Bibliothekensystems, etwa lokal auf seinem Rechner, so ist insbesondere zu beachten, daß zuerst seine Profile in das System importiert werden und dann die Auswertung eventuell vorhandener Alerting-Daten mit diesen Profilen vorgenommen wird. Nur auf diese Weise kann erreicht werden, daß er unmittelbar nach dem Einloggen über Änderungen durch das Nutzersystem informiert wird, denn die dafür notwendigen Informationen aus seinen Profilen liegen dem System erst nach dem Einloggen vor.

4.2.5 Annotationen

Annotationen zählen zu den Metadaten eines Dokumentes, denn sie beschreiben es durch Angaben, die von Nutzern gemacht wurden. Der Unterschied zu den Metadaten, wie sie

etwa im Dublin Core beschrieben werden, besteht darin, daß sie einerseits mit Zugriffsrechten geschützt werden und zumindest bisher nicht typisiert wurden. Das World-Wide Web Consortium (W3C) hat mit *Annotea* [KKPS01] eine Architektur entwickelt, die es Nutzern gestattet, HTML-Dokumente mit Hilfe eines speziellen Web-Browsers zu annotieren. Diese Annotationen werden dann entweder lokal auf dem Rechner des Nutzers oder, mit Zugriffsrechten versehen, auf einem entferntem Server abgelegt und können letztlich auch über diesen Browser abgerufen werden. Das Ändern und Löschen dieser Annotationen wird ebenfalls unterstützt. Nachfolgend wird die Architektur eines Annotationsdienstes vorgestellt, der Bestandteil der Personalisierungskomponente ist und es erlaubt, beliebige Dokumenttypen zu annotieren.

4.2.5.1 Organisation von Annotationen

Annotationen beziehen sich immer auf existierende Dokumente, befinden sich jedoch im Besitz des Nutzers, der sie geschrieben hat. Sie gehören damit nicht dem Besitzer des Dokumentes, das annotiert wurde. Die Organisation von Annotationen kann in zwei unterschiedlichen Formen erfolgen, die im folgenden erläutert werden.

Bei der ersten Variante werden alle Annotationen eines Dokumentes und eines Nutzers innerhalb dieses Dokumentes gespeichert. Der Vorteil liegt hier beim schnellen Zugriff auf das annotierte Dokument, sofern ein Nutzer nur an den Annotationen des Dokumentes interessiert ist, die ein einziger Nutzer verfaßt hat. Der Nachteil dieser Organisationsform besteht zum einen in den u. U. sehr hohen Speicheranforderungen, da für jeden Nutzer, der ein Dokument annotiert hat, eine eigene Version dieses Dokumentes gehalten werden muß. Zum anderen ist es nicht möglich, die Annotationen mehrerer Nutzer eines Dokumentes gleichzeitig zu präsentieren. Dies könnte zwar ermöglicht werden, indem beispielsweise zusätzlich ein Dokument mit den Annotationen aller Nutzer bzw. aller Kombinationen von Nutzern gespeichert wird. Der Speicherverbrauch wird dadurch aber nochmals deutlich gesteigert, so daß diese Organisationsform als nicht praktikabel einzustufen ist.

Ein zweite Variante organisiert die Annotationen getrennt von den dazugehörigen Dokumenten. Dabei wird zu jeder Annotation festgehalten, von welchem Nutzer sie stammt, zu welchem Dokument sie gehört und welche Nutzer darauf zugreifen dürfen. Beim Zugriff auf ein annotiertes Dokument werden die Annotationen, an denen der Nutzer interessiert ist, in des geforderte Dokument eingefügt und zurückgeliefert. Die Annotationen können jedoch auch separat angefordert werden. Werden auf eine Anfrage hin die Annotationen in das dazugehörige Dokument eingefügt, dann erhöht sich die Antwortzeit des Systems, was beim Zurückliefern separater Annotationen dagegen nicht vorkommt. Trotzdem sind beide Varianten sinnvoll. Zusammenfassend läßt sich festhalten, daß die getrennte Verwaltung von Dokumenten und Annotationen einfach, effizient und nichtredundant ist und es ermöglicht, daß ein Dokument mit den Annotationen mehrerer Nutzer bzw. daß separate Annotationen unter Beachtung der Zugriffsrechte auf eine Anfrage hin zurückgeliefert werden können.

4.2.5.2 Aufbau und Funktionsweise des Dienstes

Die Aufgaben des Annotationsdienstes untergliedern sich in des Verwalten von Annotationen und in die Bereitstellung eines Suchdienstes, mit dem die Suche über den Annotationen ermöglicht wird. Der Aufbau des Dienstes gestaltet sich wie in Abbildung 4.10 dargestellt. Da Annotationen auch Volltexte sind, über denen gesucht wird, muß neben den Annotationen selbst auch ein Volltextindex aufgebaut und verwaltet werden. Das Einfügen neuer Annotationen und die Indizierung dieser wird durch den Annotation-Handler vorgenommen, ebenso das Update bzw. Entfernen von Annotationen und die entsprechende Anpassung des Volltextindex. Aufgrund der Einschränkung, nach der nur Nutzer Annotationen zu den Dokumenten machen dürfen, auf die sie zugriffsberechtigt sind, müssen beim Anlegen einer Annotation die Rechte des entsprechenden Dokumentes überprüft werden, was durch eine Anfrage an den Replikationsdienst geschieht, der Bestandteil des Virtuellen Dokumentenservers ist. Eine weitere Aufgabe des Annotation-Handlers besteht im lesenden Zugriff auf Annotationen und die dazugehörigen Metadaten (Autor, zugehöriges Dokument) mittels einer ID. Bevor eine Annotation zurückgegeben wird, wird überprüft, ob der Zugriff darauf berechtigt ist.

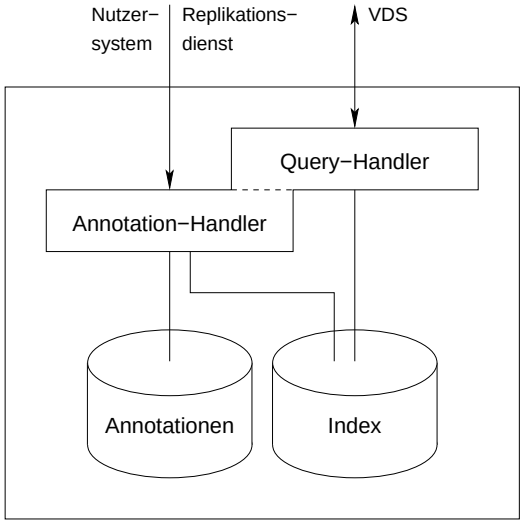

Abbildung 4.10: Aufbau des Annotationsdienstes

Die Aufgabe, Anfragen nach Annotationen zu beantworten, ist Aufgabe des Query-Handlers. Dabei kann in den Anfragen, die vom VDS gestellt werden, sowohl nach den Metadaten der Annotationen (Autor, zugehöriges Dokument) als auch in deren Volltext gesucht werden. Die Volltextsuche wird mit Hilfe des Index durchgeführt, der durch den Annotation-Handler verwaltet wird. Die Suche nach Metadaten und der Zugriff auf Annotationen infolge einer Anfrage bearbeitet der Query-Server durch Nutzung des Annotation-Handlers (gestrichelte Linie in der Abbildung), der, wie oben beschrieben, den Zugriff kapselt.

4.2.5.3 Bemerkung

Obwohl eine Annotation immer zu einem Dokument gehört, ist sie nicht Eigentum des Nutzers, der auch das Dokument besitzt, sondern sie gehört dem, der sie verfaßt hat. Dies bedeutet insbesondere, daß beim Löschen eines Dokumentes durch dessen Besitzer höchstens seine Annotationen entfernt werden dürfen. Die Annotationen anderer Nutzer zu diesem Dokument bleiben bestehen und können nur durch deren Autoren gelöscht werden.

4.2.6 Kooperation

Unter Kooperation soll im Zusammenhang mit personalisierten Digitalen Bibliotheken die Möglichkeit betrachtet werden, die es verschiedenen Nutzern erlaubt, bestimmte Ressourcen untereinander zu teilen. Zu solchen Ressourcen zählen neben lokalen Dokumenten und Annotationen auch Suchergebnisse und Suchanfragen, die sich, wie in Abschnitt 4.3.1 beschrieben, speichern lassen. Da diese Ressourcen von dafür zuständigen Diensten verwaltet werden, etwa der Annotationen vom Annotationsdienst, und diese Dienste den Zugriff auf diese Ressourcen unter Beachtung der jeweils gesetzten Zugriffsrechte bereitstellen, ist ein separater Dienst zum Zwecke der Kooperation nicht notwendig. Im Allgemeinen bedeutet dies, daß zusätzliche Aufgaben, in diesem Fall die Bereitstellung einer Kooperationsmöglichkeit, von bereits bestehenden Diensten übernommen wird. Damit steigt zwar die Auslastung bestimmter Dienste, die Anzahl der Dienste bleibt dafür überschaubar, im Gegensatz zur Bereitstellung immer weiterer Dienste.

Um Kooperationsmöglichkeiten bereitzustellen, muß zwischen verschiedenen Rechten differenziert werden, mit denen Ressourcen ausgestattet werden können. Der folgende Abschnitt gibt deshalb einen Überblick über mögliche Rechte.

4.2.6.1 Modellierung von Zugriffsrechten

Grundsätzlich kann in Digitalen Bibliotheken zwischen den folgenden Rechten unterschieden werden:

Leserecht Jeder Nutzer, der das Leserecht einer Ressource erhält, kann diese einsehen. Die Art dieser Einsicht hängt von der Art der Ressource ab. Ein Dokument etwa kann mit Hilfe eines Betrachters gelesen oder auf dem Drucker ausgegeben werden, Annotationen können dagegen separat oder innerhalb eines Dokumentes eingesehen werden. Alle in dieser Arbeit vorgestellten Komponente machen ausschließlich von diesem Recht Gebrauch.

Schreibrecht Das Schreibrecht erweitert das Leserecht, d. h. ein Nutzer, der dieses Recht für eine bestimmte Ressource erhält, kann diese nicht nur ändern, sondern auch lesend darauf zugreifen.

Replikationsrecht Dieses Recht regelt, ob Dokumente von externen Servern lokal, also auf dem eigenen Bibliotheksserver gespeichert werden dürfen.

Weitergabe von Rechten Anderen Nutzern kann nur derjenige Rechte auf Ressourcen erteilen, der entweder der Besitzer dieser Ressource ist, oder aber das Recht zur

Weitergabe von Rechten besitzt. Dieses Recht berechtigt nicht nur zur Weitergabe sondern auch zum Entzug bereits erteilter Rechte.

Zeitlich begrenztes Recht Mit dieser Variante können beliebige Rechte vergeben werden, die lediglich für einen bestimmten Zeitraum gültig sind. Nach Ablauf dieses Zeitraumes werden die so erteilten Rechte wieder entzogen.

Rechte können an verschiedene Nutzer vergeben werden. Diese können dabei in verschiedenen Formen adressiert werden:

Nutzergruppen Mehrere Nutzer werden zu einer Gruppe zusammengefaßt. Erhält eine Nutzergruppe ein bestimmtes Recht, so erhalten alle Nutzer in der Gruppe dieses Recht.

Nutzerrollen Ein Nutzer kann verschiedene Rollen spielen. Wird ein Zugriffsrecht an eine Nutzerrolle vergeben, so können die Nutzer dann Gebrauch von diesem Recht machen, wenn sie sich in dieser Rolle befinden. Ein Nutzer kann zu einem Zeitpunkt immer nur in eine einzige Rolle spielen.

4.2.7 Anbindung klassischer Bibliotheksdienste

Neben den für Digitale Bibliotheken typischen Diensten sollen auch solche eingebunden werden, die den traditionellen Bibliotheken entstammen. Zu diesen Diensten werden insbesondere gezählt:

- die Suche in verschiedenen Katalogen (Metadatensuche)
- die Vorbestellung von Titeln (Bücher, Zeitschriften etc.)
- das Abwickeln der Fernleihe
- die Verlängerung der Ausleihfrist

Diese Dienste werden nachfolgend kurz charakterisiert und deren Einbindung in die Personalisierungskomponente und damit in die Digitale Bibliothek beschrieben.

4.2.7.1 Katalogsuche

Die Suche in Bibliothekskatalogen umfaßt im Allgemeinen die Suche in analogen Katalogen, etwa den sogenannten Handzettelkästen, und die Suche auf digitalen Medien. Die Suche in den analogen Handzetteln kann nicht von einem digitalen Dienst aus erfolgen. Da aber die Inhalte der Handzettelkästen immer weiter digitalisiert werden, werden diese Daten auch auf dem digitalen Weg zugänglich.

Ein Bibliothekskatalog, der in digitaler Form vorliegt, auf dem also elektronisch gesucht werden kann, kann als eine Art Lokaler Dokumentenserver betrachtet werden, auf dem sich ausschließlich Metadaten befinden. Um Nutzeranfragen, die an den Virtuellen Dokumentenserver gestellt werden, auch auf vorhandene Bibliothekskataloge auszuweiten, müssen diese beim VDS angemeldet werden. Da sich die Formate der dieser Bibliothekskataloge stark voneinander unterscheiden können, muß zwischen dem VDS und den

Katalogen der Föderationsdienst eingeschaltet werden. Dieser vermittelt, wie in Abbildung 4.11 dargestellt, Anfragen des VDS an den Katalog und gibt dessen Anfrageergebnisse in aufbereiteter Form an den VDS zurück. Die Aufbereitung ist notwendig, um dem VDS die Ergebnisse in einem Format zu liefern, das dieser versteht und weiterverwenden kann, beispielsweise um die Anfrageergebnisse der beteiligten Dokumentenserver zu mischen.

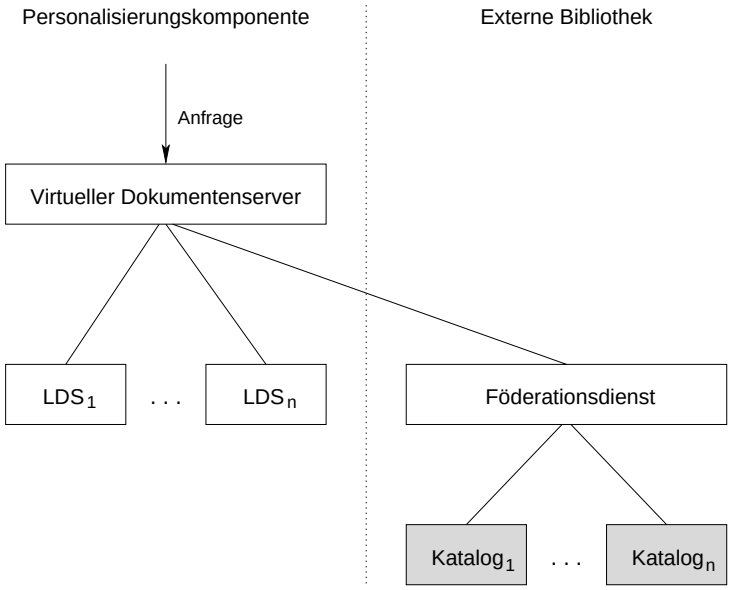

Abbildung 4.11: Anbindung von Bibliothekskatalogen

4.2.7.2 Andere Dienste

Unter “anderen Diensten” werden die Vorbestellung, die Fernleihe und die Verlängerung von Ausleihfristen verstanden. Die Abwicklung dieser Dienste wird vom Klassischen Bibliotheksdienst übernommen, der Teil der Personalisierungskomponente ist. Er besteht, wie in Abbildung 4.12 gezeigt, aus einer Leih- und einer Fristverwaltung, welche die Leihdaten verwalten.

Bei der Vorbestellung von Titeln, sowohl in der “hauseigenen” Bibliothek, als auch in externen Bibliotheken via Fernleihe, werden die Metadaten dieser Titel vom Nutzer am Nutzersystem eingegeben. Diese Daten werden daraufhin an den Klassischen Bibliotheksdienst gesendet und von dort an den jeweiligen Bibliotheksserver. Leiht ein Nutzer einen Titel in der Bibliothek aus, so gibt es zwei Möglichkeiten, diesen Vorgang im System festzuhalten:

1. Der Nutzer gibt selber ein, wann er welchen Titel ausgeliehen hat und wann die Leihfrist abläuft.

2. Das System der Bibliothek, bei der das Buch oder die Zeitschrift ausgeliehen wurde, informiert die Personalisierungskomponente, oder genauer den Klassischen Bibliotheksdienst.

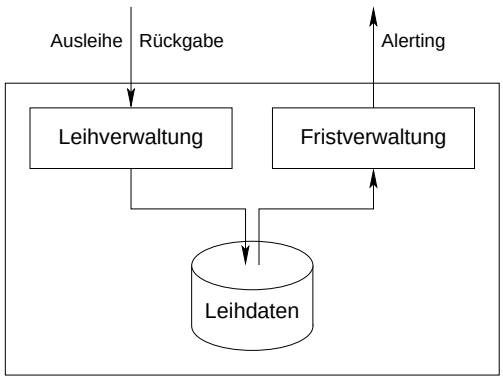

Abbildung 4.12: Aufbau des Klassischen Bibliotheksdienstes

Der Klassischen Bibliotheksdienst (KBD) speichert die Daten zunächst ab, unabhängig davon, auf welchem Wege sie geliefert wurden. Im Anschluß daran folgt die Verwertung dieser Daten, um den Nutzer entweder daran zu erinnern, daß seine Leihfrist innerhalb der nächsten Tage ausläuft oder die Ausleihfrist des Titels automatisch zu verlängern, sofern dies möglich ist.[13]

Benachrichtigung Um den Nutzer darüber zu informieren, daß die Leihfrist eines von ihm ausgeliehenen Titels innerhalb weniger Tage ausläuft, wird zunächst der Benachrichtigungsdienst aus Abschnitt 4.2.4 derart erweitert, daß er in der Lage ist, auszuliefernde Einträge der Alerting-Datenbank erst zu einem bestimmten Zeitpunkt auszuliefern. Dies kann auf einfache Weise erreicht werden, indem die Einträge, die nicht unmittelbar ausgeliefert werden sollen, immer den Zeitpunkt ihrer Auslieferung enthalten.

Leiht ein Nutzer nun beispielsweise ein Buch aus, dann wird der Alerting-DB des Benachrichtigungsdienstes ein Eintrag hinzugefügt, der vor Ablauf der Leihfrist dieses Buches an den Ausleiher gesendet wird und einen Hinweis auf das Ablaufen der Frist enthält.

Automatische Verlängerung der Leihfrist Die automatische Verlängerung von Leihfristen wird dadurch erreicht, daß sich der Klassische Bibliotheksdienst merkt, wann die Leihfrist abläuft. Vor dem Ablauf der Frist versucht er dann, diese zu verlängern, indem er die dafür notwendigen Daten (Nutzername, Paßwort[14], Titel) an den Server der Bibliothek sendet. Gelingt der Versuch, d. h. wurde der Titel erfolgreich verlängert, dann wird wiederum der Zeit gespeichert, zu dem die neue Frist ausläuft, um sie dann erneut zu verlängern. Kann der Titel dagegen nicht verlängert werden, dann wird in die Alerting-DB ein Eintrag eingefügt, der unmittelbar auszuliefern ist und den Nutzer über den Ablauf der Frist erinnert.

[13] Die Ausleihfrist eines Titels kann nur dann verlängert werden, wenn dieser nicht vorbestellt wurde.

[14] Nutzername und Paßwort, die zum Zugriff auf den Bibliotheksserver benötigt werden, werden zusammen mit den Nutzereigenschaften im Nutzerprofil abgespeichert.

Gibt der Nutzer sein ausgeliehenes Buch zurück, dann muß der Klassische Bibliotheksdienst auch darüber informiert werden. Dies kann entweder durch den Nutzer selber geschehen, oder automatisch über den Server der Bibliothek.

4.3 Sekundäre Dienste

Zu den sekundären Diensten der hier vorgestellten Personalisierungskomponente zählen

- die Nachbearbeitung von Anfrageergebnissen
- das Hinzufügen von Dokumentenservern, auf die Suchanfragen ausgedehnt werden können
- die Erstellung und Verwaltung von Handapparaten
- die sogenannte wiederholte Suche

die in den folgenden Abschnitten diskutiert, konzipiert und dem Digitalen Bibliothekensystem zugefügt werden sollen.

4.3.1 Anfragenachbearbeitung

Unter der Anfragenachverarbeitung soll die nachträgliche Bearbeitung von Anfrageergebnissen verstanden werden. Sie wird dann durchgeführt, wenn der Virtuelle Dokumentenserver ein Anfrageergebnis auf eine Anfrage hin zurückliefert. Die Anfragenachbearbeitung wird, wie auch die Vorverarbeitung der Anfragen, vom Anfragedienst der Personalisierungskomponente ausgeführt, deren Aufbau in Abbildung 4.5 auf Seite 31 zu finden ist. Der Anfragedienst erhält Anfrageergebnisse vom VDS, bearbeitet sie innerhalb des Postprozessors, und liefert sie über die Anfrageverwaltung an das Nutzersystem. Zur Aufgabe der Anfragenachbearbeitung gehören die folgenden Teile:

Inkrementelle Anfragen Anfragen können nach der Berechnung der Anfrageergebnisse nachbearbeitet und erneut an das System gestellt werden, wobei zur Nachbearbeitung sowohl das Verfeinern als auch das Verallgemeinern der Anfrage zählen. Diese Art der Nachbearbeitung wird separat in Abschnitt 4.3.4 beschrieben.

Tupeloperationen Die Anfrageergebnisse werden in Form von Tupeln durch den VDS zurückgeliefert. Diese Tupel können dann verändert werden, etwa um nachträglich Tupel zu löschen oder deren Reihenfolge zu ändern.

Layout Anfrageergebnisse werden dem Nutzer präsentiert. Um die Art dieser Präsentation zu steuern, können Attribute wie Farben oder Fonteigenschaften gesetzt werden.

Relevance Feedback Nutzer können bewerten, welche Güte ein Anfrageergebnis bezüglich ihrer gestellten Anfrage aufweist. Infolge dieser Bewertung werden Gewichte angepaßt, die in zukünftige Anfragen einfließen. Die Änderung dieser Gewichte soll bewirken, daß neue Anfragen Ergebnisse liefern, die nach Meinung des

Nutzers qualitativ hochwertiger sind. Da die Einarbeitung von Relevance Feedback-Möglichkeiten in die Personalisierungskomponente den Rahmen dieser Arbeit übersteigen würde, soll hier darauf verzichtet und lediglich auf Literatur zu diesem Gebiet verwiesen werden, etwa [BYRN99].

4.3.1.1 Tupeloperationen

Die Ergebnisliste, die vom Virtuellen Dokumentenservers berechnet und zurückgeliefert wird, besteht nach [Tit99] aus

- den Attributen (bzw. Funktionen und arithmetischen Operationen), nach denen in der Anfrage gesucht wurde
- Dokumentenschlüsseln (Handles), die zum Zugriff auf die jeweiligen Dokumente benötigt werden
- Informationen zum Ranking

Bei der Durchführung von Operationen auf der Ergebnisliste werden nun insbesondere die enthaltenen Attribute bearbeitet, Ranking-Informationen und Dokumentenschlüssel werden nicht geändert, sondern höchstenfalls entfernt. Es können die folgenden Operationen ausgeführt werden:

Filterung Mittels Filterung können Tupel aus dem Ergebnis entfernt werden. Diese Operation erscheint vor allem dann sinnvoll, wenn Nutzer nicht das komplette Ergebnis einer Anfrage sehen wollen, sondern immer nur Teile davon, oder etwa die Ergebnisse, die aufgrund des Rankings an oberster Stelle stehen. Tupel können auch dann herausgefiltert werden, wenn die dazugehörigen Dokumente nicht in solchen Formaten vorliegen, wie sie vom Nutzer gefordert wurden.

Sortierung Die Reihenfolge der Ergebnisliste wird durch die gestellte Anfrage bestimmt und u. U. durch den VDS beeinflußt. Trotzdem kann es notwendig sein, ein Anfrageergebnis nachträglich zu sortieren. So gibt es beispielsweise Datenbanksysteme, die ihre Ergebnisse nicht nach Attributen sortieren können, wenn diese aus Zeichenketten bestehen, die eine bestimmte Länge überschreiten. Um dieses Manko zu überbrücken, kann die gewünschte Sortierung innerhalb der Anfragenachbearbeitung nachgeholt werden.

Konvertierung Wie oben beschrieben, können Tupel aus der Ergebnisliste gelöscht werden, wenn die dazugehörigen Dokumente nicht in geforderten Formaten vorliegen. Ein andere Möglichkeit besteht darin, die Dokumente zu konvertieren. Enthält das Anfrageergebnis also Tupel (Metadaten), deren dazugehöriges Dokument nicht in einem vom Nutzer geforderten Format vorliegt, und kann dieses Dokument in eines der geforderten Formate konvertiert werden, dann bleibt das Tupel bestehen. Greift der Nutzer auf das Dokument zu, so wird es in eines der von ihm bevorzugten Formate umgewandelt und zurückgegeben.

4.3.1.2 Layout

Layouteigenschaften von Suchergebnissen, die dem Nutzer präsentiert werden, werden vom Nutzersystem ausgewertet und angezeigt. Um diese Anzeige zu steuern, wird das Anfrageergebnis des VDS (siehe oben) um einige Attribute erweitert, mit denen sich das Layout beschreiben läßt. Dazu gehören beispielsweise Fonteigenschaften und verwendete Farben, die in den Nutzerprofilen spezifiziert sind.

Wurden die Tupel eines Anfrageergebnisses bearbeitet, so wird als letzter Schritt das vom Nutzer gewünschte Layout in das Ergebnis eingearbeitet, indem die Layoutattribute gesetzt werden. Das so bearbeitete Ergebnis wird dann an die Anfrageverwaltung (siehe Abschnitt 4.3.4) zurückgeliefert, die dieses Ergebnis zwischenspeichert und an das Nutzersystem gibt, wo es dem Nutzer in der spezifizierten Form (Layout, Sortierung, etc.) präsentiert wird.

4.3.2 Hinzufügen neuer Dokumentenserver

Die Lokalen Dokumentenserver, die an den Virtuellen Dokumentenserver angeschlossen sind, decken u. U. den Informationsbedarf bestimmter Nutzer nicht ab, da beispielsweise keine Dokumente vorhanden sind, die zu seinem Forschungsgebiet gehören. In diesem Fall kann ein externer VDS aushelfen, der über einen Broker gefunden und mit Hilfe des Föderationsdienstes angesprochen werden kann. Der Nachteil dieses Verfahrens ist aber, daß der Broker anhand der Anfrage entscheidet, an welchen externen VDS eine gestellte Anfrage weitergeleitet wird. Möchte dagegen der Nutzer entscheiden, welche externen Virtuellen Dokumentenserver seine Anfragen neben den bereits eingebundenen LDS beantworten sollen, dann sind zwei andere Verfahren denkbar, die in den folgenden Abschnitten vorgestellt werden sollen. Die dabei durchzuführenden Arbeitsschritte, zu denen das An- und Abmelden von externen VDS und deren Verwaltung gehören, werden vom *VDS-Manager* durchgeführt, der in Abbildung 4.13 zu finden ist.

4.3.2.1 Anmeldung externer VDS

Um den Zugriff auf externe VDS zu ermöglichen, müssen diese zunächst angemeldet werden. Dazu muß der Nutzer insbesondere Angaben dazu machen, wie der VDS erreichbar ist, d. h. mit welchem Protokoll er kommuniziert und unter welcher Adresse er gefunden werden kann. Die bei der Anmeldung angegebenen Daten werden vom VDS-Manager gespeichert. Die eigentliche Anmeldung, bei der externe VDS dem lokalen VDS bekannt gemacht werden, nimmt der VDS-Manager ebenfalls vor, was in den beiden folgenden Abschnitten genauer erläutert wird.

4.3.2.2 Direkte Anbindung externer VDS

Bei der direkten Anbindung externer Virtueller Dokumentenserver werden diese VDSs, wie in Abbildung 4.13 dargestellt, direkt am lokalen VDS gekoppelt. Dies bedingt jedoch, daß lokaler und externe VDS mit demselben Protokoll kommunizieren und daß der lokale VDS die Form der Ergebnisse von externen VDS verstehen kann, insbesondere um die Anfrageergebnisse der beteiligten LDS und externen VDS zu mischen.

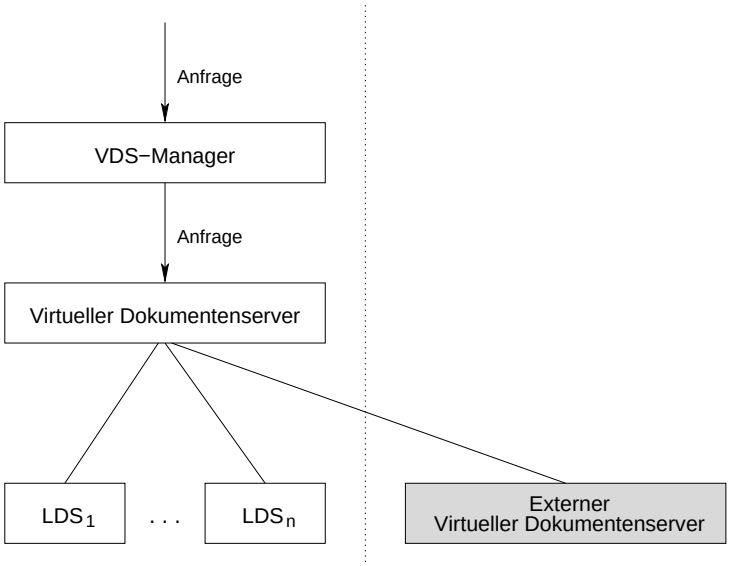

Abbildung 4.13: Direkte Anbindung externer VDS

Bei der Anmeldung von externen VDS, die direkt vom lokalen VDS angesprochen werden, wird der externe VDS bei der Ressourcenverwaltung des lokalen VDS angemeldet. Anfragen, die an den lokalen VDS gestellt werden, sendet dieser dann nicht nur an die beteiligten Lokalen Dokumentenserver, sondern auch den externen VDS, der hier wie ein LDS behandelt wird.

Um zu erreichen, daß nur die Anfragen an den externen VDS weitergeleitet werden, die von dem Nutzer stammen, der diesen VDS angemeldet hat, wird der externe VDS bei der Ressourcenverwaltung vor der Anfragebearbeitung aktiviert, danach wieder deaktiviert.[15] Somit werden Anfragen anderer Nutzer nicht auf diesen externen VDS ausgedehnt. Eine andere Möglichkeit, dies zu erreichen, besteht darin, den externen VDS dauerhaft aktiviert zu lassen, ihn aber durch Angabe der zu durchsuchenden Ressourcen[16] bei der Anfragebearbeitung anzugeben bzw. auszuschließen.

4.3.2.3 Indirekte Anbindung externer VDS

Bei der indirekten Anbindung von externen Virtuellen Dokumentenservern werden diese über einen Föderationsdienst [EHS+00] angesprochen. Anfragen an den lokalen VDS werden also nicht nur an beteiligte Lokale Dokumentenserver weitergeleitet sondern auch an den Föderationsdienst (Abbildung 4.14). Dieser leitet die Anfragen wiederum an den externen VDS weiter und liefert dessen Ergebnisse zurück. Voraussetzung für die Verwendung des Föderationsdienstes ist jedoch, daß der externe VDS diesem Föderationsdienst bekannt ist.

[15] Deaktivierte Ressourcen bleiben beim VDS angemeldet, werden jedoch nicht verwendet.

[16] Eine Anfrage kann laut [Tit99] neben dem eigentlichen Anfragestring auch eine Menge von Ressourcen enthalten, von denen die Anfrage bearbeitet werden soll.

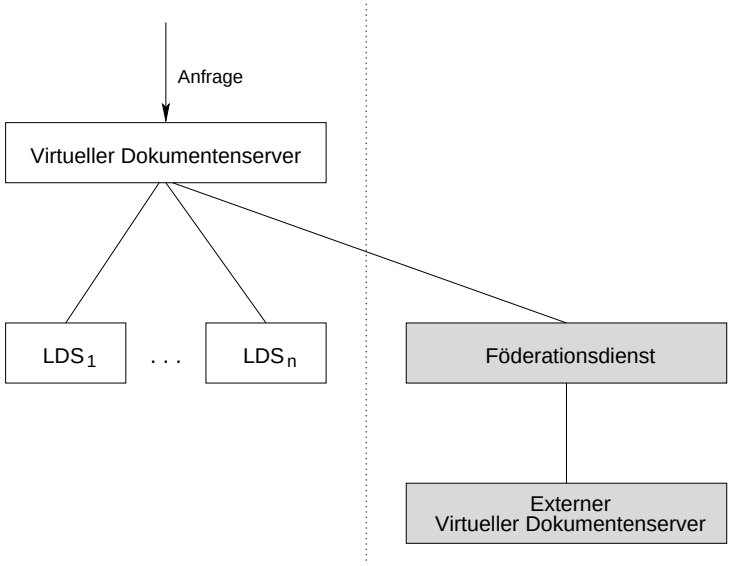

Abbildung 4.14: Indirekte Anbindung externer VDS

Der Vorteil der indirekten Anbindung liegt in der Verwendung des Föderationsdienstes, der den Zugriff auf verschiedenartigste externe Ressourcen ermöglicht. Er kann beispielsweise die verschiedenen Formate von Anfragen und Anfrageergebnissen so konvertieren, daß Dokumentenserver miteinander kommunizieren können, obwohl sie sich bei direktem Kontakt nicht verstehen würden, etwa wegen unterschiedlicher Protokolle. Der Nachteil dieser Variante ist die Tatsache, daß mit dem Föderationsdienst ein zusätzlicher Dienst in Anspruch genommen wird. Damit erhöht sich u. U. die Antwortzeit auf eine gestellte Anfrage.

4.3.2.4 Verwaltung externer VDS

Bisher wurden externe VDS immer nur für die Nutzer verwendet, die sie bei System angemeldet haben. Um diese VDS auch anderen Nutzern zugänglich zu machen, müssen sie mit Rechten versehen werden. Alle Nutzer, die die notwendigen Zugriffsrechte besitzen, können somit ebenfalls Gebrauch der externen VDS machen. Der Vorteil liegt dabei insbesondere darin, daß verschiedene Nutzer externe VDS nutzen können, ohne wissen zu müssen, wo sich diese befinden und mit welchen Protokollen darauf zugegriffen werden kann.

4.3.3 Erstellung von Handapparaten

Handapparate sind Dokumente, die aus Teilen anderer Dokumente zusammengesetzt werden. Diese Teile können Textabsätze sein, aber auch Abbildungen oder Zitate.

Um Handapparate erstellen zu können, müssen zunächst die Dokumente vom Virtuellen Dokumentenserver geholt werden, aus deren Teilen der Handapparat aufgebaut werden soll (Abbildung 4.15). Aus diesen Dokumenten müssen dann die eigentlichen Anteile

extrahiert werden, die den Handapparat bilden werden. Liegen die Dokumente, aus denen Teile extrahiert werden sollen, in einem unstrukturierten Format wie PostScript vor, dann müssen diese Dokumente zunächst inhaltlich analysiert werden, um beispielsweise herauszufinden, wo sich Absätze, Abbildungen oder Verzeichnisse befinden. Liegen die Dokumente dagegen bereits in einem beschreibenden Format vor, etwa in XML oder LaTeX, dann entfällt diese Analysephase.

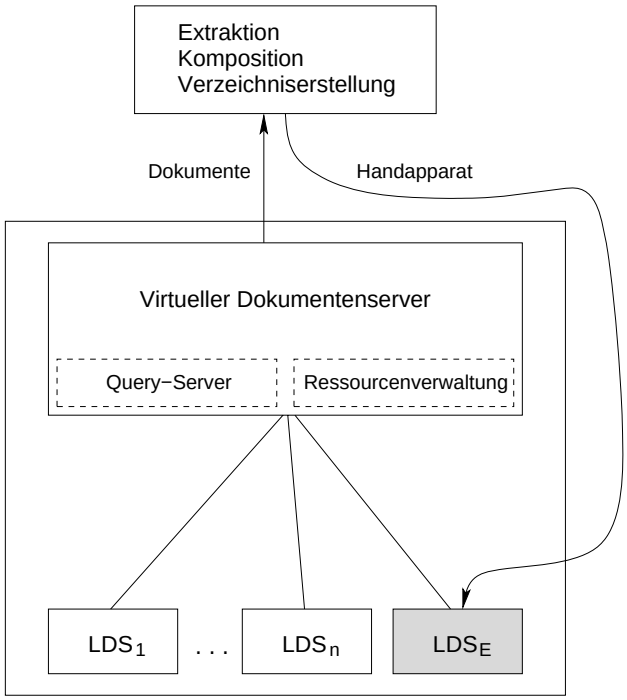

Abbildung 4.15: Erstellung eines Handapparates

Im nächsten Schritt werden die extrahierten Dokumentteile zu einem neuen physischen Dokument zusammengesetzt (Komposition). Abschließend wird das so erstellte Dokument konsistent strukturiert, d. h. vorhandene Kapitel, Abschnitte u. s. w. werden durchnumeriert und in entsprechende Verzeichnisse eingefügt, d. h. Kapitelnumerierungen werden in das Inhaltsverzeichnis eingefügt, Abbildungen in das Abbildungsverzeichnis und so weiter.

4.3.3.1 Vergabe von Zugriffsrechten

Handapparate sind im Wesentlichen “normale” Dokumente, die dementsprechend mit Zugriffsrechten versehen werden können. Da sie aber aus anderen Dokumenten bzw. Dokumentteilen zusammengesetzt sind, die ebenfalls Zugriffsrechte besitzen, können die Rechte bei Handapparaten nicht beliebig vergeben werden. Grundsätzlich gilt, daß maximal die Nutzer Zugriff auf einen Handapparat erhalten dürfen, denen auch der Zugriff auf alle Dokumente gestattet ist, aus denen der Handapparat erstellt wurde. Vergibt ein Nutzer also für einen von ihm erstellten Handapparat Zugriffsrechte, so akzeptiert das Sy-

stem diese nur, wenn sie den Rechten der verwendeten Dokumentteile entsprechen bzw. diese weiter einschränken.

4.3.3.2 Einfügen von Handapparaten in den Dokumentenbestand

Um einen erstellten Handapparat dauerhaft in der Digitalen Bibliothek zu speichern, muß der Nutzer diesen zunächst durch Angabe von Metadaten beschreiben. Im Anschluß daran wird er zusammen mit den Metadaten im erweiterten Lokalen Dokumentenserver (Abschnitt 4.2.3 engefügt und dort automatisch indexiert. Damit wird er möglich, auch nach Handapparaten zu suchen und darauf zuzugreifen.

4.3.4 “Wiederholte Suche”

Die Qualität einer Anfrage ist für den Nutzer immer erst anhand des Anfrageergebnisses ersichtlich. Stellt er nun aufgrund eines Anfrageergebnisses fest, daß seine Anfrage zu allgemein oder zu speziell war, so möchte er sie nachträglich verändern und erneut an das System stellen. Die Bereitstellung einer solchen Funktionalität wird durch die Anfrageverwaltung übernommen. Diese ermöglicht es auch, daß von Nutzern gestellte Anfragen und darauf berechnete Anfrageergebnisse dauerhaft gespeichert und miteinander verglichen werden können. Der Aufbau der Anfrageverwaltung ist in Abbildung 4.16 dargestellt. Sie besteht aus einem Query-Server, der Anfragen entgegennimmt, speichert und an den Anfragedienst aus Abschnitt 4.2.2 weiterleitet. Der zweite Bestandteil der Anfrageverwaltung ist der Resultset-Server, der die Anfrageergebnisse vom Anfragedienst entgegennimmt, speichert und an das Nutzersystem zurückliefert. Die Aufgabe des Vergleichs von Anfragen und Anfrageergebnissen wird vom Komparator vorgenommen, der dabei je nach Bedarf auf den Query- bzw. den Resultset-Server zugreift. Genauere Erläuterungen zur Funktionsweise finden sich in den Folgeabschnitten.

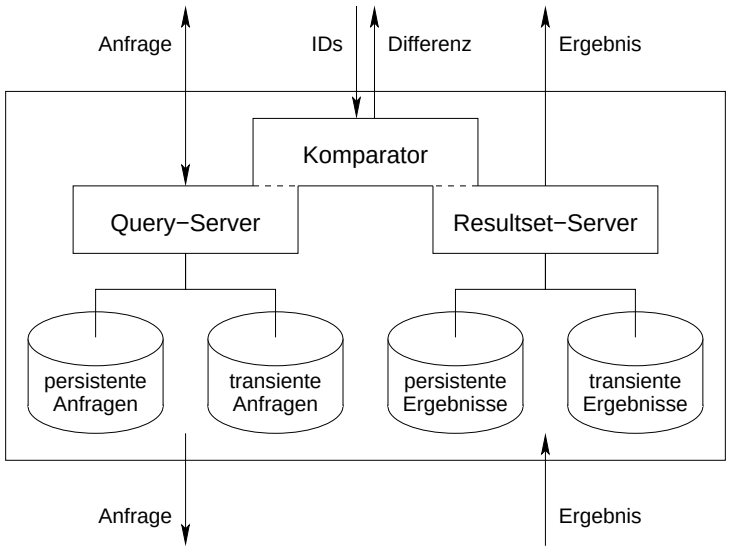

Abbildung 4.16: Aufbau der Anfrageverwaltung

4.3.4.1 Verwaltung von Anfragen

Alle Anfragen, die vom Nutzersystem an die Anfrageverwaltung gesendet werden, werden durch den Query-Server verarbeitet. Dieser speichert diese Anfragen transient mit einer eindeutigen ID ab, fügt der Anfrage diese ID hinzu und sendet sie letztlich an den Anfragedienst weiter.

Auf Verlangen des Nutzers können Anfragen zusätzlich auch persistent, also dauerhaft, gespeichert werden, wobei dies wieder unter Vergabe einer ID erfolgt. Sowohl persistente als auch transiente Anfragen können unter Angabe einer ID vom Nutzersystem angefordert werden, um sie letztlich an das System zu stellen. Der Unterschied zwischen transienten und persistenten Anfragen besteht lediglich darin, daß transiente Anfragen nach Ablauf einer gewissen Zeit aus dem Speicher verdrängt werden, persistente Anfragen dagegen nur durch den Nutzer gelöscht werden können.

Um einen Schutz von gespeicherten Anfragen zu erreichen, werden transiente und persistente Anfragen mit Zugriffsrechten versehen. Der Zugriff darauf wird nur den Nutzern gestattet, die laut Rechtevergabe zum Kreis der Zugriffsberechtigten gehören. Somit können Nutzer nicht nur ihre eigenen Anfragen betrachten, vergleichen und an das System absetzen, sondern auch die anderer Nutzer. Das Ziel der Kooperation aus Abschnitt 3.3.6 wird damit auch für Anfragen verwirklicht.

4.3.4.2 Verwaltung von Anfrageergebnissen

Alle Anfrageergebnisse werden vom Virtuellen Dokumentenserver an die Anfrageverwaltung geliefert. Dort werden sie vom Resultset-Server zunächst transient mit einer ID gespeichert. In der Anfrage ist u. a. die ID der Anfrage enthalten, die zu diesem Anfrageergebnis geführt hat. Die ID des Anfrageergebnisses wird ebenfalls dem Anfrageergebnis hinzugefügt, so daß diese nun die ID der dazugehörigen Anfrage und die eigene ID enthält. Das so erweiterte Anfrageergebnis wird dann an das Nutzersystem übergeben.

Aufgrund der IDs, die im Anfrageergebnis enthalten sind, kann das Nutzersystem die ID des Anfrageergebnis extrahieren und abspeichern, so daß der Nutzer durch die Ergebnisse seiner Anfragen navigieren kann. Durch Extrahierung der Anfrage-ID ist es außerdem möglich, auf die Anfrage zuzugreifen, die zum vorliegenden Ergebnis geführt hat. Stellt der Nutzer also anhand eines Anfrageergebnisses fest, daß er seine Anfrage nachträglich ändern muß, um zu einem zufriedenstellenden Ergebnis zu gelangen, dann wird die Anfrage über die Anfrage-ID geholt, um sie nachzubearbeiten und erneut an das System zu stellen.

Möchte ein Nutzer das vorliegende Anfrageergebnis dauerhaft speichern, so wird für dieses Ergebnis vom Resultset-Server eine ID vergeben unter der er es schließlich in den Bestand der persistenten Anfrageergebnisse aufnimmt. Der Unterschied zwischen transienten und persistenten Anfrageergebnissen ist, genau wie bei den Anfragen, daß transiente nach einer gewissen Zeit durch das System aus dem Speicher entfernt werden, persistente dagegen nur durch den Nutzer gelöscht werden können.

Auch Anfrageergebnisse werden mit Zugriffsrechten versehen, um gewährleisten zu können, daß nur berechtigte Nutzer darauf zugreifen können. Abgesehen vom Schutz vor unerlaubten Zugriffen können Nutzer somit auch kooperieren, indem sie Zugriff auf fremde Ergebnisse erhalten.

4.3.4.3 Vergleichsoperationen

Gestellte Anfragen und dazu berechnete Anfrageergebnisse werden immer transient, auf Wunsch des Nutzers zusätzlich auch persistent gehalten. Um nun beispielsweise das Ergebnis zweier Anfragen miteinander vergleichen zu können, wird der sogenannte Komparator verwendet, der Bestandteil der Anfrageverwaltung ist. Unter Angabe zweier IDs, die sowohl zu Anfragen als auch zu Anfrageergebnissen gehören können, greift der Komparator zunächst über Query- bzw. Resultset-Server auf die Anfragen bzw. die Anfrageergebnisse zu. Handelt es sich bei den zu vergleichenden Objekten um Anfragen, so werden diese an das System abgesetzt, um die Anfrageergebnisse zu berechnen. Im entscheidenden Schritt berechnet der Komparator die Differenz der Ergebnisse und liefert sie zurück.

4.4 Zusammenspiel der Dienste

In den vorigen Abschnitten dieses Kapitels wurden hauptsächlich einzelne Dienste vorgestellt. Um das Zusammenspiel eines Großteils dieser Dienste zu beschreiben, soll nun geschildert werden, in welcher Weise Nutzeranfragen nach Metadaten von den verschiedenen Diensten der Personalisierungskomponente bearbeitet werden.

Nutzeranfragen werden immer über das Nutzersystem gestellt und danach zur Bearbeitung an die Personalisierungskomponente übergeben. Diese bearbeitet die Anfrage nach dem folgenden Schema, das in Abbildung 4.17 illustriert ist:

1. Die Anfrage wird von der Anfrageverwaltung gespeichert. Dadurch wird es möglich, wie an Abschnitt 4.3.4 beschrieben, Anfragen beispielsweise inkrementell zu stellen.

2. Der Anfragedienst speichert die Anfrage zunächst ebenfalls ab, um bei verschiedenen Anfragen eines Nutzers zu erkennen, welche Interessen er hat. Im nächsten Schritt wird die Anfrage vom Anfragedienst mit Hilfe der Profildaten des Nutzers so erweitert, daß sich seine generellen Interessen in der Anfrage wiederspiegeln und seine Einstellungen enthalten sind.

3. Die erweiterte Anfrage wird vom Virtuellen Dokumentenserver weiterbearbeitet, der sie zum einen auf seine Lokalen Dokumentenserver und zum anderen an den Erweiterten LDS verteilt. Die Anfrage wird außerdem über einen Föderationsdienst an externe VDS und an Katalogserver traditioneller Bibliotheken gesendet.

4. Die Anfrageergebnisse, die der VDS von seinen Lokalen Dokumentenservern, dem erweiterten LDS, externen VDS und Katalogservern erhält, werden gemischt und an den Anfragedienst weitergereicht.

5. Der Anfragedienst führt eine Nachverarbeitung des Anfrageergebnisses wieder mit Hilfe der Profildaten vom Profildienst durch. Dabei werden beispielsweise Ergebnistupel gefiltert oder Layoutattribute festgelegt, die bei der Anzeige des Ergebnisses im Nutzersystem verwendet werden.

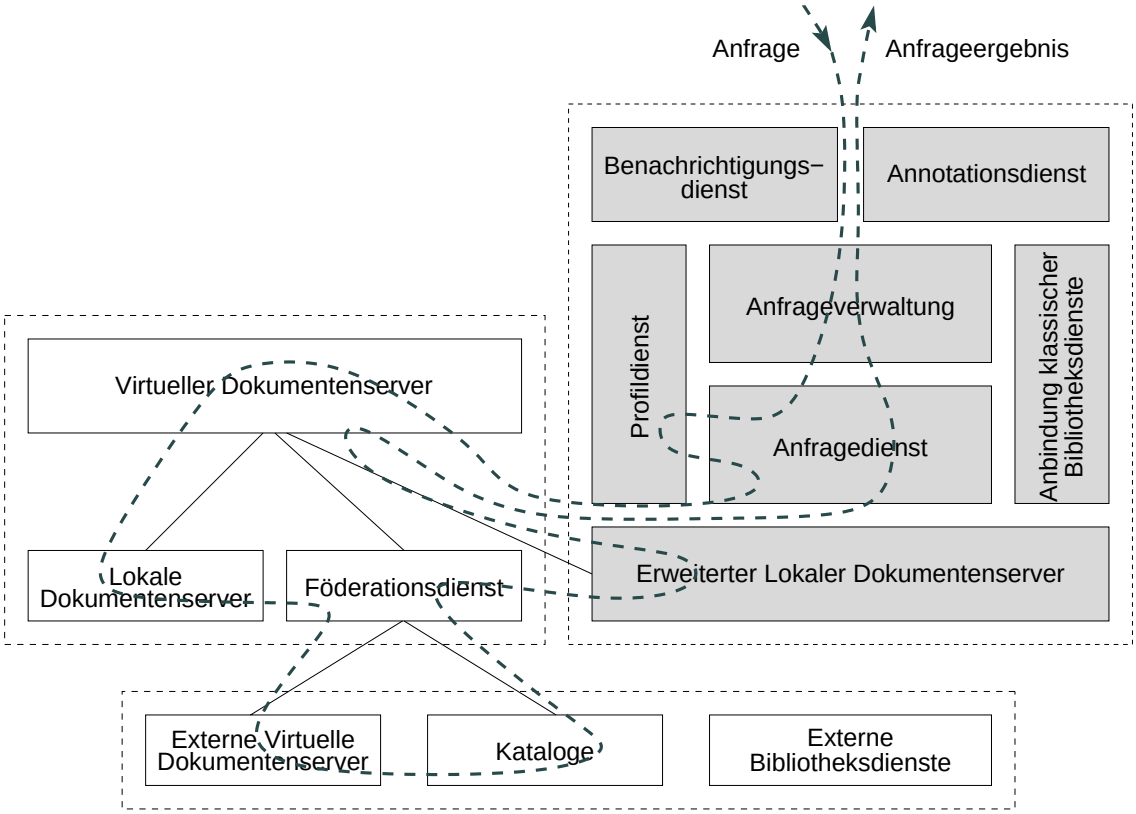

Abbildung 4.17: Anfragebearbeitung der Personalisierungskomponente

6. Das nachbearbeitete Ergebnis wird nun von der Anfrageverwaltung gespeichert, wodurch dem Nutzer etwa das Navigieren durch seine verschiedenen Anfrageergebnisse oder der Vergleich zwischen diesem und später berechneten Anfrageergebnissen ermöglicht wird.

7. Das so bearbeitet Ergebnis wird schließlich an das Nutzersystem geliefert, von dem aus die Anfrage gestellt wurde, die zur Berechnung von diesem Ergebnis geführt hat.

Das Beispiel der Anfragebearbeitung zeigt, daß durch die Aufteilung der verschiedensten Aufgaben an unterschiedliche Dienste zu einer gewissen Beherrschbarkeit des Gesamtsystems führt. Würde das komplette System aus lediglich einer Komponente bestehen, wäre es für den Programmierer aufgrund seiner Komplexität nur schwer überschaubar. Es wäre außerdem äußerst anfällig, da ein Ausfall immer das Ausfallen des kompletten Systems bedeuten würde. Der Ausfall einer Komponente des hier vorgestellten Systems bedeutet dagegen lediglich, daß das Gesamtsystem zwar nur eingeschränkt arbeitsfähig ist, aber nicht zwangsläufig einen Totalausfall impliziert.

4.5 Zusammenfassung

Im hier vorliegenden Kapitel wurden verschiedene Dienste vorgestellt, deren Einsatz zur Umsetzung von Personalisierungskonzepten in Digitalen Bibliotheken dient. Diese Dienste, die in Abbildung 4.2 auf Seite 19 graphisch zusammengefaßt sind, erfüllen die Anforderungen, die in Kapitel 3 an sie gestellt wurden, wobei der Fokus auf den primären Diensten lag, insbesondere auf dem Profil- und dem Benachrichtigungsdienst. Dies läßt insbesondere Raum für eine weitergehende Konzeption der sekundären Dienste. Änderungen im Bereich der primären Dienste sind aber ebenfalls denkbar. Beispielsweise wäre es möglich, Anfrageverwaltung und Anfragedienst zu einem einzigen Dienst zu verschmelzen, was einen geringeren Kommunikationsaufwand zwischen diesen Diensten bedeutet, aber auch mit der Erweiterung des Aufgabenbereiches des Dienstes einhergeht, der aus der Verschmelzung der beiden entsteht.

Kapitel 5

Prototypische Implementierung

Ein Teil dieser Diplomarbeit besteht aus der Implementierung einiger Teile der Architektur, die im vorigen Kapitel vorgestellt wurde. Diese Implementierung umfaßt einen globalen Benachrichtigungsdienst, einen dafür notwendigen Profildienst und eine Nutzerschnittstelle, über die der Profildienst konfiguriert werden kann. Die Grundlage dieser Implementierung ist ein Virtueller Dokumentenserver, der in den Projekten [ZZZ+01] und [HD01] entstanden ist. Die hier entwickelten Dienste sollen so daran gekoppelt werden, so daß alle Alerting-Daten des Benachrichtigungsdienstes in den VDS eingefügt werden und damit für Suchenanfragen an den VDS zu Verfügung stehen.

In den folgenden Abschnitten wird zunächst beschrieben, welche Techniken und Protokolle zur Realisierung der Implementierung eingesetzt werden können. Anschließend erfolgt eine Auswahl dieser Techniken und Protokolle, die konkret zur Implementierung verwendet wurden. Danach wird die Funktionsweise der implementierten Dienste erläutert und mit denen aus Kapitel 4 in Bezug gesetzt.

5.1 Grundlegende Konzepte

Zur Realisierung eines Digitalen Bibliothekensystems werden verschiedene Konzepte benötigt. Da einem solchen System meist eine Dienstearchitektur zugrunde liegt und die beteiligten Dienste verteilt vorliegen, sind Konzepte zur Programmierung verteilter Systeme notwendig. Weil in Digitalen Bibliothekensystemen außerdem riesige Datenmengen verwalten werden müssen und es dementsprechend sinnvoll erscheint, für diese Verwaltung Datenbanksysteme einzusetzen, müssen Schnittstellen zu Datenbanken verwendet werden. Ein Teil der Kommunikation zwischen verschiedenen Diensten geschieht unter Verwendung des XML-Formates, so daß Konzepte zur Verarbeitung von XML-Dokumenten benötigt werden.

Aus dies Gründen werden nun Konzepte vorgestellt, die skizzieren, wie verteilte Anwendungen implementiert, Datenbanken angebunden und XML-Dokumente verarbeitet werden können. Konzepte zur Entwicklung von Web-Schnittstellen, über die der Nutzer auf eine Digitale Bibliothek zugreifen kann, werden außerdem vorgestellt.

5.1.1 Entwicklung verteilter Anwendungen

Die in dieser Arbeit vorgestellte Architektur ist eine sogenannte Dienste-Architektur. Dies bedeutet, daß das System aus mehreren einzelnen Komponenten, den Diensten, besteht. Solche Dienste können allesamt auf einem Server laufen, aber auch weit verstreut sein, beispielsweise zur Verteilung der Systemlast. Besonders bei einer verteilten Architektur spielt die Kommunikation zwischen den Diensten eine wichtige Rolle, da sie u. a. Einfluß auf die Laufzeit des Gesamtsystems hat. Dauert etwa der Datenaustausch zwischen Virtuellem Dokumentenserver und den Lokalen Dokumentenservern zu lange, so wirkt sich dies negativ auf die Antwortzeit des VDS aus, selbst wenn die Bearbeitung einer Anfrage von VDS und den beteiligten LDS für sich jeweils sehr schnell ablaufen.

Zur Realisierung der Kommunikation zwischen verteilten Diensten existieren mehrere Möglichkeiten. Ein Auszug derjenigen, die sich etabliert haben bzw. voraussichtlich in naher Zukunft etablieren werden, soll nun vorgestellt werden.

5.1.1.1 Low-Level-Netzwerkprogrammierung

Unter der diesem Begriff verbirgt sich die Programmierung auf einem relativ wenig abstrahierten Niveau. Dabei wird beispielsweise direkt mit sogenannten Sockets gearbeitet, zwischen denen sich eine Verbindung herstellen läßt, um darüber Daten zu übertragen. Die Verwendung von Protokollen wie HTTP oder FTP kann ebenfalls zur Low-Level-Netzwerkprogrammierung gezählt werden. Diese Protokolle bestehen aus einem relativ kleinen Befehlssatz, der die Datenübertragung kapselt und somit schon deutlich vereinfacht.

Der Vorteil dieser Art der Programmierung ist eine hohe Performanz. Dagegen steht jedoch die Tatsache, daß der Aufwand für den Programmierer relativ hoch ist. Außerdem ist es nicht möglich, Objekte im Sinne der Objektorientierten Programmierung zu übertragen, da keine Abbildung zwischen verschiedenen Programmiersprachen vorgenommen wird. Eine Einführung in die Netzwerkprogrammierung mit Java kann etwa in [CW99] gefunden werden, eine gute Referenz bietet [Har00].

5.1.1.2 CORBA

Die *Common Object Request Broker Architecture* (kurz CORBA) [cor98] ermöglicht die Programmierung von verteilten Anwendungen, die objektorientiert implementiert wurden. Die Idee von CORBA ist es, Methoden von Objekten aufzurufen, unabhängig davon, auf welchem Rechner sich diese befinden und in welcher (objektorientierten) Programmiersprache sie implementiert wurden. Die Kommunikation wird dabei von drei Komponenten geregelt:

Stub Ein Stub ist ein Objekt einer Klasse, die eine vorgegebene Schnittstelle implementiert und auf dem Client abgelegt wird. Als Client wird der Rechner bezeichnet, von

dem aus eine entfernte Methode aufgerufen wird.

Skeleton Ein Skeleton ist auch einer Objekt einer Klasse, die eine vorgegebene Schnittstelle implementiert. Im Unterschied zu Stubs werden Skeletons auf dem Server abgelegt, also auf einem der entfernten Rechner, auf dem sich die aufzurufenden Methoden der Objekte befinden.

Object Request Broker (ORB) Die Verbindung zwischen Stubs und Skeletons wird vom ORB hergestellt. Liegen Stub- und Skeleton-Objekt in jeweils verschiedenen Programmiersprachen vor, dann nimmt der ORB eine Umwandlung vor, damit das Skeleton-Objekt den Methodenaufruf und das Stub-Objekt die eventuelle Antwort versteht.

Zur Realisierung von entfernten Methodenaufrufen mit Hilfe von CORBA werden die Schnittstellen der implementierten Klassen sprachunabhängig mit Hilfe der IDL (Interface Definition Language) beschrieben. Ein IDL-Compiler generiert aus dieser Schnittstellenbeschreibung Stubs und Skeletons, die auf dem Client bzw. dem Server hinterlegt werden. Der Aufruf einer Methode wird nun wie folgt bearbeitet (siehe Abb. 5.1):

1. Die aufzurufende Methode wird mit ihren Parametern vom Client über den Stub an den ORB übergeben.

2. Der ORB sucht den Server, auf dem sich das Objekt befindet, dessen Methode aufgerufen werden soll.

3. Auf dem entfernten Rechner wird der Methodenaufruf vom ORB über das Skeleton an das Objekt weitergereicht, und dort schließlich bearbeitet.

4. Nach Abarbeitung der Methode wird eine eventuelles Ergebnis, das auch eine Ausnahme (Exception) sein kann, in entgegengesetzter Richtung über Skeleton, ORB und Stub an die aufrufende Methode zurückgegeben.

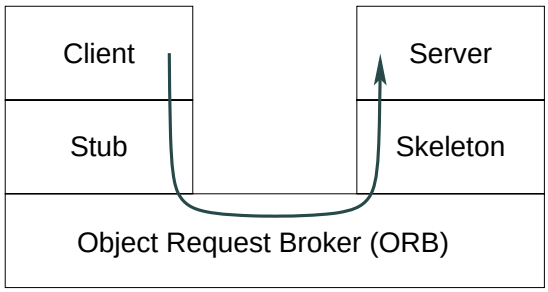

Abbildung 5.1: Ablauf eines entfernten Methodenaufrufs in CORBA

CORBA ist eine sehr mächtige Architektur, insbesondere weil sie weitestgehend sprachunabhängig ist und im Vergleich zur Low-Level Netzwerkprogrammierung ein sehr hohes Abstraktionsniveau besitzt. Leider erfordert sie einen relativ hohen infrastrukturellen Aufwand. Der Aufwand zur Anbindung von CORBA an andere Architekturen, wie etwa

DCOM (Distributed Component Object Model, Microsoft), ist ebenfalls sehr hoch. Diese Nachteile werden in der folgenden Variante zur Programmierung verteilter Anwendungen beseitigt, jedoch auf Kosten der Sprachunabhängigkeit.

5.1.1.3 Remote Method Invocation (RMI)

RMI ist, ähnlich wie CORBA, eine konkrete Realisierung zum Aufruf entfernter Methoden. Im Unterschied zu CORBA ist RMI nicht sprachunabhängig, sondern speziell für Java konzipiert worden. Auch der Ablauf beim Aufruf einer entfernten Methode kann mit dem von CORBA verglichen werden, wobei jedoch die folgenden Unterschiede bestehen:

- Beim Zugriff auf die Methode eines entfernten Objektes (Abb. 5.2) wird nicht direkt auf das entfernte Objekt, sondern eine Kopie (Proxy-Objekt) davon zugegriffen, die sich auf dem Client befindet. Ein solches Proxy-Objekt wandelt die übergebenen Argumente in einen serialisierbaren Datenstrom um und leitet diesen über einen Transportkanal an des “echte” entfernte Objekt auf dem Server weiter.
- Die Rückgabewerte einer entfernten Methode werden ebenfalls nicht direkt an das Objekt auf dem Client geliefert, sondern an ein Proxy-Objekt davon, welches sich auf dem Server befindet. Dieses Proxy-Objekt wandelt diesen Wert in einen serialisierbaren Datenstrom um und sendet diesen an den Client.
- Es muß keine Umwandlung von Objekten zwischen verschiedenen Programmiersprachen vorgenommen werden

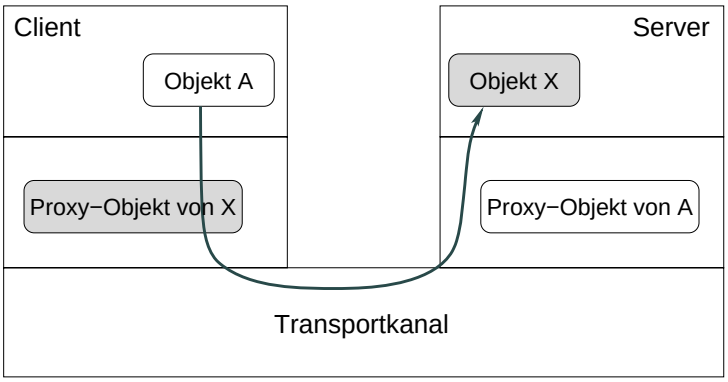

Abbildung 5.2: Aufruf einer entfernten Methode mit RMI

RMI besitzt wie CORBA ein sehr hohes Abstraktionsniveau, ist sehr leicht einzusetzen und ist bereits Standard im JDK. Damit besitzt jeder Java-Programmierer von vornherein eine RMI-Implementierung. Ein Nachteil gegenüber CORBA besteht jedoch in der festen Bindung an die Programmiersprache Java.

5.1.1.4 Simple Object Access Protocol (SOAP)

Das *Simple Object Access Protocol* ist ein Protokoll, welches auf anderen Protokollen wie HTTP (Hyper Text Transfer Protocol) oder SMTP (Simple Mail Transfer Protocol)

aufbaut. Kern des Protokolls ist der Datenaustausch von Softwarekomponenten, unabhängig davon, wo sich diese befinden und in welcher Sprache sie implementiert wurden. Das Format eines Datenaustausches mittels SOAP ist mit XML ein Textformat, dessen Struktur von einem vorgegebenen Schema bestimmt wird.

Zum Aufruf einer entfernten Methode werden bei SOAP die folgenden Schritte durchgeführt:

1. Generieren des Methodenaufrufs in Form eines XML-Dokumentes.

2. Senden des XML-Dokumentes, etwa mit Hilfe eines HTTP-Requests (POST), an die Softwarekomponente, deren Methode aufgerufen werden soll.

3. Empfangen des XML-Dokumentes von der entfernten Komponente, deren Methode aufgerufen werden soll.

4. Auswertung des XML-Dokumentes und Durchführung der Anweisungen, die in diesem Dokument enthalten sind. Damit wird die Methode (mit den entsprechenden Parametern) aufgerufen, die im XML-Dokument angegeben wurde.

Die Rückgabe eines Ergebnisses, das bei der Ausführung der entfernten Methode berechnet wurde, erfolgt analog, d. h. es wird in Form eines XML-Dokumentes geliefert, beispielsweise via HTTP, aus dem XML-Dokument extrahiert und verarbeitet.

SOAP ist ein relativ einfaches Protokoll, dessen Anwendung zwar schon deutlich abstrakter als die Low-Level Netzwerkprogrammierung ist, jedoch nicht den Abstraktionsgrad von CORBA oder RMI erreicht. Es vereint im Grunde Vor- und Nachteile von Low-Level-Netzwerkprogrammierung, CORBA und RMI in sich, da es einfach zu handhaben ist, verteiltes Programmieren ermöglicht und unabhängig von der Programmiersprache ist. Das geringe Abstraktionsniveau kann sowohl als Vorteil ausgelegt werden, da ein geringer Overhead anfällt, sowie als Nachteil, da der Aufwand für den Programmierer höher ist als beispielsweise bei RMI.

Eine Kurzeinführung in SOAP gibt [Mod01], eine Referenz mit vielen Beispielen kann in [GHMN01] gefunden werden.

5.1.2 Anbindung von Datenbanken

Digitale Bibliotheken sind Systeme, die, wie schon in Abschnitt 3.2 erwähnt, große Datenmengen verarbeiten und verwalten müssen. Datenbanksysteme wurden u. a. für die Verwaltung großer Datenmengen geschaffen, so daß es nahe liegt, Digitale Bibliotheken auf der Basis von Datenbanksystemen aufzubauen. Nachfolgend sollen darum die wesentlichen Schnittstellen umrissen werden, die den Zugriff auf Datenbanken aus einem Programm heraus ermöglichen.

5.1.2.1 Call Level Interface (CLI)

Die Nutzung einer *Call-Schnittstelle* (Call Level Interface) beruht auf speziellen Bibliotheken einer konkreten Programmiersprache, die den Datenbankzugriff kapseln. Um aus einem Programm heraus beispielsweise eine Anfrage an eine Datenbank zu stellen und

das Anfrageergebnis auswerten zu können, müssen lediglich die von der Bibliothek bereitgestellten Funktionen bzw. Methoden aufgerufen werden.

Der Vorteil dieser Schnittstellen liegt in der Einfachheit ihrer Anwendung, da nur bestehende Methoden aufgerufen werden müssen. Nachteilig ist jedoch, daß Syntaxfehler der im Programm gestellten Datenbankanfragen erst zur Laufzeit des Programms gefunden werden.

5.1.2.2 Java Database Connectivity (JDBC)

JDBC ist ein Vertreter der Call-Schnittstellen, der für Java geschaffen wurde (siehe etwa [WFC+99], [SS00] oder [Mey00]). Teil eines jeden JDBC-Treibers sind Java-Klassen und -schnittstellen, die den Zugriff auf Datenbanken kapseln bzw. vorschreiben. Die Typen der verschiedenen JDBC-Treiber werden danach unterteilt, ob sie:

1. den Datenbankzugriff über einen ODBC-Treiber abwickeln (Typ-1-Treiber)

2. in nativem Code (Typ 2) oder in Java geschrieben wurden und sich einer Middleware zwischen Datenbank-Client und -server bedienen (Typ 3)

3. in Java geschrieben und ohne nativen Code arbeiten (Typ 4).

Der Einsatz von JDBC hat, ähnlich wie Java selbst, in den vergangenen Jahren immer stärker zugenommen, was vor allem mit der einfachen Anwendung und der breiten Unterstützung dieser Schnittstelle in diversen Datenbanksystemen zu begründen ist.

5.1.2.3 Embedded SQL

Bei der eingebetteten Programmierung mit SQL (Embedded SQL) werden SQL-Anweisungen im Quelltext einer konkreten Programmiersprache benutzt. Diese Anweisungen sind getrennt von den Anweisungen der Programmiersprache zu betrachten und werden durch spezielle Schlüsselwörter hervorgehoben. Um eine Interaktion zwischen den SQL-Anweisungen und den Anweisungen der Programmiersprache herzustellen, werden spezielle Variablen benutzt, sogenannte Host-Variablen. Auf diese Variablen kann sowohl innerhalb der SQL-Statements als auch vom Quelltext der Programmiersprache aus zugegriffen werden.

Um den mit SQL-Statements versehenen Quelltext zu übersetzen, muß er zunächst von einen Präcompiler vorverarbeitet werden. Ein solcher Präcompiler ersetzt alle SQL-Anweisungen durch Aufrufe von speziellen Funktionen bzw. Methoden eines Call Level Interfaces (siehe Abschnitt 5.1.2.1), die in derselben Sprache geschrieben wurden, wie der zu übersetzende Quelltext. Das Ergebnis des Präkompilierens wird abschließend von einem Compiler in ausführbaren Code übersetzt.

Ein Vorteil von Embedded SQL ist, daß Syntaxfehler in den SQL-Aufrufen bereits zum Zeitpunkt der Vorübersetzung erkannt werden. Nachteilig wirkt sich jedoch die Tatsache aus, daß mit einer zweigeteilten Syntax programmiert werden muß: eine für die Programmiersprache und eine für die SQL-Anweisungen.

5.1.2.4 Embedded SQL für Java (SQLJ)

SQLJ ist ein Bündel aus drei Paketen, mit deren Hilfe Zugriffe aus Java-Programmen auf Datenbanken realisiert werden können. Diese Pakete wurden wie folgt gestaltet:

Teil 1: Embedded SQL in Java Innerhalb von Java-Methoden können SQL-Statements angegeben werden, die wie im letzten Abschnitt erwähnt, hervorgehoben werden müssen, um sie syntaktisch von den Java-Anweisungen unterscheiden zu können. Der so erstellte Quellcode wird von einem Präcompiler vorübersetzt, wobei die SQL-Anweisungen durch JDBC-Anweisungen ersetzt werden. Das Ergebnis des Präcompilers kann dann mit einem Java-Compiler zu Byte-Code übersetzt werden.

Teil 2: Java-Methoden als Stored Procedures Java-Methoden, die mittels eingebetteter Programmierung (SQLJ) erstellt wurden, können im Datenbanksystem hinterlegt und wie eine Stored Procedure innerhalb einer Datenbankanfrage aufgerufen werden.

Teil 3: Java-Klassen als SQL-Typen Mit Hilfe dieses Teils von SQLJ können automatisch Abbildungen von Java-Klassen auf Datenbankrelationen erstellt werden. Dazu werden einerseits die Java-Klassen definiert und andererseits die Relationen, in denen Objekte dieser Klassen gespeichert werden sollen. Das Mapping übernimmt SQLJ, so daß es nicht mehr die Aufgabe des Programmierers ist, das Speichern von Objekten in einer Datenbank zu implementieren.

Weitere Erläuterungen und insbesondere Beispiele für SQLJ aber auch für JDBC sind in [SS00] zu finden.

5.1.3 Bereitstellung und Anbindung von Web-Schnittstellen

Web-Schnittstellen (Web-Formulare) werden mit Hilfe von HTML bereitgestellt. Der Inhalt eines solchen Formulars wird mittels Java-Script ausgelesen und an den Web-Server gesendet. Für die Auswertung und Weiterverarbeitung solcher Formulare gibt es zwei grundlegende Ansätze:

1. CGI-Programmierung

2. Erweiterung von Standard-HTML.

Bei der Nutzung des Common Gateway Interface (CGI) werden Programme in einer beliebigen Programmiersprache auf dem Web-Server hinterlegt. Beim Aufruf eines solchen Programms können die Werte des ursprünglichen Web-Formulars ausgelesen und schließlich verarbeitet, etwa in einer Datenbank abgelegt werden. Darüberhinaus besteht die Möglichkeit, HTML-Seiten dynamisch zu generieren und an den Web-Client zurückzusenden.

Die Basis von CGI ist immer eine Programmiersprache, die in der Lage ist, selbst oder durch Zuhilfenahme von Erweiterungspaketen Anfragen aus dem Internet, also via HTTP,

zu verstehen und Antworten zu generieren. Für die CGI-Programmierung ist die Verwendung von Perl besonders verbreitet, aber auch Java findet durch den Einsatz sogenannter Servlets[1] [Cow01] immer mehr Akzeptanz.

Eine andere Variante, Web-Schnittstellen anzubinden, besteht darin, Standard-HTML um neue Tags zu erweitern. Diese speziellen Tags gestatten beispielsweise den Zugriff auf Datenbanken oder auch das dynamische Generieren neuer HTML-Seiten. Ein Vertreter dieser Variante ist Ingweb [ing], mit dem der Zugriff auf Ingres-Datenbanken ermöglicht wird. Ein anderer und sehr viel mächtigerer Vertreter sind die sogenannten Java Server Pages (JSP) [Llo01], mit deren Hilfe vollständige Java-Programme in HTML-Seiten eingebettet werden können. Diese Programme werden beim ersten Aufruf der Seite, in die sie eingebettet sind, in die oben erwähnten Java Servlets übersetzt und schließlich ausgeführt.

5.1.4 Verarbeitung von XML-Dokumenten

XML-Dokumente werden mit Hilfe sogenannter XML-Prozessoren verarbeitet, die in Form von Bibliotheken für diverse Programmiersprachen existieren. Diese Prozessoren verfolgen dabei zwei Ansätze zum Verarbeiten von XML: das *Document Object Model* und das *Small API for XML*. Beide sollen nachfolgend skizziert werden. Eine ausführlichere Darstellung dieser Ansätze findet sich in [Zei00].

5.1.4.1 Document Object Model (DOM)

Bei der Verwendung von DOM wird das zu verarbeitende XML-Dokument komplett eingelesen und im Hauptspeicher eine baumartige Struktur aufgebaut, die dieses Dokument repräsentiert. Diese Struktur wird als *Document Object Model* bezeichnet. In diesem Modell werden Elemente, Attribute u. s. w. des XML-Dokumentes durch Baumknoten dargestellt. Um das ursprüngliche Dokument auszuwerten oder zu bearbeiten wird der DOM-Baum traversiert.

Diese Methode ist zwar sehr komfortabel, aber leider auch sehr zeitintensiv, da immer erst das komplette Dokument eingelesen werden muß, bis es verarbeitet werden kann. Da auch der komplette DOM-Baum aus dem XML-Dokument im Hauptspeicher gehalten wird, ist dieses Vorgehen insbesondere bei großen Dokumenten äußerst speicherintensiv.

5.1.4.2 Small API for XML (SAX)

Die Grundlage von SAX ist die Ereignisorientierung. Dabei wird ein zu analysierendes XML-Dokument teilweise eingelesen, was einem Ereignis entspricht, und daraufhin eine Methode aufgerufen, die zu diesem Ereignis paßt. Wird also beispielsweise ein Element eingelesen, dann wird im Anschluß daran eine Methode zur Verarbeitung von Elementen aufgerufen. Gleiches gilt für alle anderen XML-Bestandteile, etwa Kommentare, Elementwerte oder Prozessoranweisungen.

SAX erscheint zwar u. U. nicht so intuitiv wie DOM, bietet aber den großen Vorteil, daß XML-Dokumente speicherschonend und sehr schnell verarbeitet werden können. Die

[1] In Java geschriebene CGI-Programme heißen Servlets, wobei sich der Name *Servlet* aus der Tatsache ableitet, daß diese Programme auf dem Web-Server ablaufen.

Nachteile von DOM treten hierbei also nicht auf, wodurch sich SAX besonders für die Verarbeitung umfangreicher Dokumente eignet.

5.2 “Wahl der Mittel”

Von den in den vorigen Abschnitten vorgestellten Konzepten, Schnittstellen u. s. w. wurden zur Implementierung des Benachrichtigungs- und Profildienstes die folgenden ausgewählt:

Programmiersprache Als Programmiersprache wurde Java gewählt, da diese Sprache und vor allem die dazugehörige Bibliothek, das Java Development Kit (JDK) ein schnelles Prototyping ermöglichen und die Umsetzung verteilter Anwendungen relativ einfach ist. Zusätzlich bietet die Objektorientierung und die selbstständige Freispeicherverwaltung der Sprache ein hohes Abstraktionsniveau. Die Verwendung eines Exception-Mechanismus legt letztlich die Basis für sicherer Programmieren, da so einfach und adäquat auf auftretende Fehler reagiert werden kann.

Datenbankschnittstelle Zur Anbindung eines Datenbanksystems wurde JDBC eingesetzt. Da Grund dafür ist zum einen die relativ einfache Anwendung dieser Schnittstelle, zum anderen wird sie momentan von den Datenbankherstellern besser unterstützt als etwa SQLJ.

Realisierung der Diensteverteilung Die Verteilung der Dienste, die hier aus der Anbindung des Hermes-Servers besteht, wurde mittels RMI vorgenommen. Gründe dafür sind, daß der Hermes-Server von außen nur via RMI erreichbar ist. Zum anderen wurde der Dienst, der vom Hermes-Server bereitgestellt wird, vollständig in Java implementiert.

XML-Prozessor Für die Verarbeitung von XML-Dokumenten kommt ein SAX-Parser zum Einsatz. Weil die zu parsenden Dokumente sehr groß sind, würde die Verwendung eines DOM-Parsers nicht nur zu einer deutlich längeren Laufzeit beim Parsen der Dokumente bedeuten, sondern auch einen größeren Speicherbedarf.

Web-Schnittstelle Die Implementierung einer Web-Schnittstelle erfolgte mit Hilfe von Java-Servlets. Dies liegt daran, daß die komplette Implementierung des Prototyps in Java erfolgte und somit ein Umstieg auf eine andere Sprache zur Realisierung einer solchen Schnittstelle wenig Sinn macht. Da Servlets für den Programmierer wie ein “normales Programm” aussehen und sich Fehler im Programm leichter finden lassen, setzen sich Servlets gegenüber der Verwendung von Java Server Pages durch.

5.3 Implementierung

Im Rahmen dieser Diplomarbeit wurden ein Profil- und ein Benachrichtigungsdienst prototypisch implementiert. Die Kernpunkte dieser Implementierung werden in den beiden folgenden Abschnitten vorgestellt.

5.3.1 Profildienst

Der Profildienst verwaltet zwei Arten von Profilen: konkrete und generalisierte Profile. Konkrete Profile werden vom Nutzer erstellt und beinhalten Metadaten, die charakterisieren, welche Dokumente ihn interessieren bzw. über welche Dokumente er benachrichtigt werden möchte, sobald sie in einen Dokumentenbestand eingefügt werden. Generalisierte Profile entstehen dagegen aus den konkreten Profilen durch Anwendung des Generalisierungsalgorithmus aus Abschnitt 4.2.1.3.

Beide Profilarten werden innerhalb einer relationalen Datenbank gespeichert und können die folgenden Attributtypen beinhalten:

- Autor eines Artikels
- Titel eines Artikel oder einer Zeitschrift
- Wörter, die im Titel oder in der Zusammenfassung eines Artikels auftreten.

Während konkrete Profile ausschließlich lokal gespeichert und verwaltet werden, werden die generalisierten Profile zusätzlich auf einem entfernten Alerting-Server (Hermes) RMI abgelegt, was im nächsten Abschnitt beschrieben wird.

5.3.2 Benachrichtigungsdienst

Der implementierte Benachrichtigungsdienst führt ein globales Alerting entsprechend Abschnitt 4.2.4 durch. Dies bedeutet, daß der hier implementierte Dienst an einen externen Alerting-Service[2] angebunden wird und somit über Änderungen an externen Dokumentenbeständen informieren kann. Darüberhinaus wird der der externe Dienst entlastet, da die Nutzerprofile, die auf dem selbst entwickelten Benachrichtigungsdienst abgelegt werden, wenn möglich zusammengefaßt bzw. generalisiert und erst dann extern gespeichert werden. Auf diese Weise werden weniger Profile auf dem externen Alerting-Dienst abgelegt. Die eigentliche Entlastung ergibt sich daraus, daß auf dem Hermes-Server die Alerting-Daten gegen weniger Profile geparst werden müssen.

Um den implementierten Benachrichtigungsdienstes anzuwenden, muß er zunächst vom Nutzer konfiguriert werden. Dies geschieht mit Hilfe des Profildienstes, mit dem Nutzer Profile anlegen und editieren können. Solche Profile enthalten neben den Nutzerinteressen (etwa bestimmten Autoren) auch eine Angabe darüber, wie oft der Benachrichtigungsdienst den Nutzer über Änderungen an Dokumentenkollektionen informieren soll. Die so angelegten Profile werden dann generalisiert und mit Hilfe von RMI auf dem externen Alerting-Service (Hermes) gespeichert.

Der Vorgang der Benachrichtigung untergliedert sich in die folgenden Schritte (siehe auch Abb. 5.3):

1. Übermitteln von Metadaten des Hermes-Servers an den hier vorgestellten Alerting-Dienst[3]
2. Vorverarbeitung dieser Metadaten

[2] Dabei handelt es sich um den in Anschnitt 4.2.4.1 vorgestellten Dienst HERMES der FU Berlin.
[3] Ein Beispiel einer E-Mail, wie sie vom Hermes versendet werden, ist in Abschnitt A.4 zu finden.

3. Parsen/Strukturieren der Metadaten

4. Filtern der strukturierten Metadaten gegen die Profile

5. Generieren von Nachrichten aus den Metadaten

6. Auslieferung der generierten Nachrichten

Wird der Hermes-Server von Informationsanbietern (etwa Verlagen) darüber informiert, daß ihren Datenbeständen neue Dokumente hinzugefügt wurden, dann wird überprüft, auf welche Profile diese Dokumente passen. Passen diese Dokumente auf ein generalisiertes Profil des hier vorgestellten Alerting-Dienstes, so werden diesem die Metadaten der neuen Dokumente via E-Mail im XML-Format übermittelt. Da das Format dieser Metadaten z. T. vom Schema abweicht, welches durch das erweiterte Majour-Format[4] vorgegeben ist, werden diese Daten zunächst durch Einsatz eines XSLT-Prozessors vorverarbeitet. Das Ergebnis dieser Vorverarbeitung sind XML-Dokumente, deren Struktur einem einheitlichen Schema folgt.

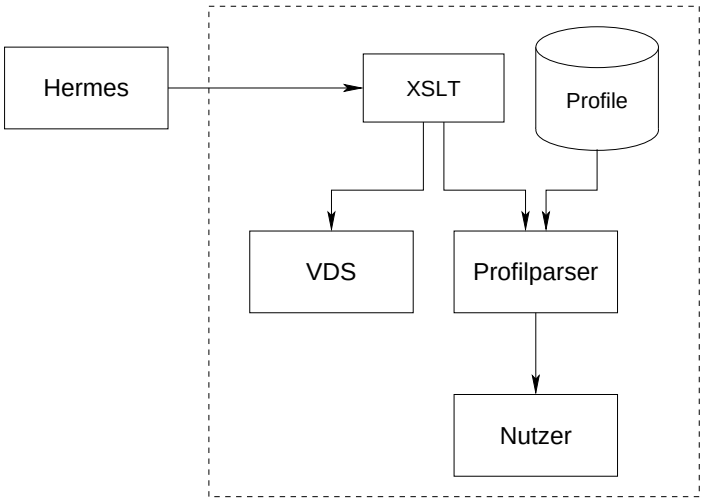

Abbildung 5.3: Ablauf einer Benachrichtigung

Im nächsten Schritt werden die vorverarbeiteten XML-Dokumente mit Hilfe eines SAX-Parsers analysiert, d. h. es werden die Titel, die Autoren, der Abstract u. s. w. der Dokumente herausgefiltert, die durch diese XML-Daten beschrieben werden, um sie im Anschluß daran in einer relationalen Datenbank abzulegen.

Danach wird das sogenannte Profile-Filtering durchgeführt, das heißt es wird überprüft, auf welche konkreten Profile die im vorigen Schritt gewonnen Metadaten passen. Dazu werden jedoch nur Profile überprüft, die in jenes generalisierte Profil eingeflossen sind, auf das die vom Hermes-Server übermittelten Daten passen. In Anhang A.2 werden

[4] Das Schema dieser Metadaten entspricht einem erweiterten Majour-Format [Maj00], mit dem es, im Gegensatz zum reinen Majour, möglich ist, nicht nur einzelne Dokumente sondern auch Dokumentenkollektionen zu beschrieben.

verschiedene Techniken des Profile-Filterings vorgestellt. Das hier verwendete Verfahren beruht jedoch ausschließlich auf der Verwendung einer relationalen Datenbank. Bei diesem Verfahren wird zu jedem Bestandteil eines Profils getestet, ob seine Bestandteile (Autor, Titel, Abstract etc.) als Substring in den Metadaten auftreten. Treten alle Bestandteile eines Profils auf, dann gilt dieses Profil als passend bezüglich der Metadaten. Anhand des folgenden Anfragerumpfes kann beispielsweise überprüft werden, ob das Wort "data" unabhängig von Groß- bzw. Kleinschreibung innerhalb eines Abstracts in einem Dokument auftritt:

```
SELECT ...
FROM   ...
WHERE UCASE(abstract) LIKE %DATA%
```

Alle Metadaten, die auf ein konkretes Profil passen, werden im nächsten Schritt an die Besitzer dieser Profile gesendet, jedoch erst nach Ablauf der Zeit (täglich, wöchentlich, monatlich), die sie beim Erstellen des Profils angegeben haben. Im letzten Schritt werden diese Metadaten einem Lokalen Dokumentenserver abgelegt, der an den Virtuellen Dokumentenserver von [HD01] gekoppelt ist. Damit werden diese Daten nicht nur zum Alerting benutzt, sondern zusätzlich für weitere Suchanfragen an den Virtuellen Dokumentenserver bereitgestellt.

5.3.3 Prototypischer Charakter der Implementierung

Die hier vorgestellte Implementierung ist von prototypischem Charakter, d. h. sie erfüllt nicht alle Anforderungen aus Kapitel 3 und sie stellt auch keine Referenzimplementierung der Architektur aus Kapitel 4 dar. Die Vereinfachungen des Prototyps umfassen die folgenden Punkte:

Nutzersystem Das System, durch das der Nutzer Zugang zur Digitalen Bibliothek erhält, ist ein Web-Browser. Dieser wird mit Hilfe von Java-Script und, wie bereits beschrieben, mit Java-Servlets gesteuert.

Profildienst Der Profildienst führt keine Trennung der Profildaten in Nutzereigenschaften, Nutzerinteressen und Nutzereinstellungen durch, wie sie in Abschnitt 4.2.1 auf Seite 20 eingeführt wurde. Der Gründe dafür liegen zum einen darin, daß nur Nutzer des Fachbereiches Zugriff auf den Prototyp haben. Deren Nutzereigenschaften, wie die Nutzerkennung oder das Paßwort, werden bereits durch das Network Information System (NIS) verwaltet. Zum anderen verwaltet der Profilmanager keine gesonderten Einstellungen. Darüberhinaus ist die Menge der Attributtypen auf den Autor, den Titel, den Abstract und den Zeitschriftennamen eines Artikels beschränkt. Diese Einschränkung wird durch den Hermes-Server vorgegeben, da die Profile auf diesem nur die vier angegebenen Typen besitzen können. Würde der Prototyp weitere Typen zulassen, so ließen sich die Profile nicht mehr auf die von Hermes abbilden.

Benachrichtigungsdienst Der Benachrichtigungsdienst des Prototyps führt ein globales Alerting durch. Ein lokales Alerting, mit dem Nutzer über Änderungen an den

Dokumenten informiert werden können, die lokal verwaltet[5] werden, wird nicht durchgeführt.

Alle Fähigkeiten des Profil- und des Benachrichtigungsdienstes, die hier nicht ausgeschlossen wurden, werden vom Prototypen bereitgestellt.

5.3.4 Ausblick

Die Umsetzung des hier vorgestellten Prototyps wurde vollständig in Java realisiert. Leider weisen in Java geschriebene Programme den z. T. gravierenden Nachteil auf, daß sie relativ langsam laufen. Der Umstieg auf andere Programmiersprachen, die besseres Laufzeitverhalten aufweisen, etwa C oder C++, sollte somit zukünftig zumindest dort angestrebt werden, wo massenhaft Daten verarbeitet werden, etwa beim Parsen von XML-Dokumenten.

Das Versenden von Alerting-Daten des Hermes-Server erfolgt derzeit via E-Mails. Für die Zukunft wurde diebezüglich geplant, diese Art der Benachrichtigung durch die Verwendung von SOAP[6] zu ersetzen. Längerfristig wird eine Anbindung des Hermes-Service an beliebige Bibliotheken geplant. Ob diese Anbindung in der Form gestaltet wird, wie in der prototypischen Implementierung, oder ob auf den jeweiligen Bibliotheksservern eine Kopie des Servers eingespielt wird, ist bisher jedoch noch unklar.

Das Parsen von Alerting-Daten gegen Profile basiert derzeit auf einer Substring-Suche. Um die Ergebnisse diese Parsings zu verbessern, müssen andere Kriterien herangezogen werden. Kandidaten dafür sind die aus dem Information-Retrieval bekannten Methoden des Stemmings, die Benutzung von Synonymen oder eine tolerante Suche, bei der Begriffe auch dann als zueinander passend erkannt werden, wenn sie in der Schreibweise leicht voneinander abweichen.

[5] Unter dem Verwalten von lokalen Dokumenten wird hier das Speichern, Durchführen von Updates und Löschen von Dokumenten eines Nutzers mit Hilfe des Bibliothekensystems verstanden.

[6] Vorstellbar ist auch die Verwendung eines Protokolls, das SOAP ähnlich ist, etwa XML-RPC [LJD01] oder XMI.

Kapitel 6

Abschließende Bemerkungen

Das Ziel dieser Diplomarbeit war die Ausarbeitung von Konzepten, durch die Personalisierung im Umfeld Digitaler Bibliotheken realisiert werden kann. Diese Konzepte waren weiterhin konzeptionell in Form einer Personalisierungskomponente, die aus verschiedenen Diensten besteht, in eine bereits bestehende Architektur einer Digitalen Bibliothek einzuarbeiten. Um die Realisierbarkeit dieser Dienste zu zeigen, wurde eine prototypische Implementierung einiger ausgewählter Dienste angefertigt.

In diesem Kapitel werden die wichtigsten Fakten zusammengefaßt, die beim Lösen der gestellten Aufgabe zusammengetragen wurden. Ein Ausblick auf weitere Aufgaben, die sich auf der Basis dieser Arbeit stellen lassen, runden das Kapitel ab.

6.1 Zusammenfassung

Der Kern der Arbeit beginnt in Kapitel 3, in dem zunächst allgemeine Anforderungen spezifiziert wurden, die die Personalisierungskomponente erfüllen muß. Im Anschluß daran wurden die Fähigkeiten dieser Komponente erläutert, die im Rahmen der Personalisierung denkbar bzw. wünschenswert sind. In Kapitel 4 wurden dann einzelne Dienste konzipiert, die die Anforderungen aus Kapitel 3 umsetzen und die in ihrer Gesamtheit die Personalisierungskomponente bilden. Zu diesen Diensten zählen:

Benachrichtigungsdienst Der Benachrichtigungsdienst informiert Nutzer immer dann, wenn sich Änderungen an Dokumentenbeständen ergeben, die für sie von Interesse sind. Derartige Änderungen beinhalten beispielsweise das Hinzufügen von Dokumenten zu externen Dokumentenbeständen, etwa von Verlagen. Änderungen können aber auch das Aktualisieren oder Löschen von Dokumenten sein, die mit Hilfe der Digitalen Bibliothek verwaltet, also im erweiterten Lokalen Dokumentenserver (siehe unten) gehalten werden.

Anfrageverwaltung/Anfragedienst Die Aufgabe der Anfrageverwaltung ist das Speichern von Nutzeranfragen und den dazu berechneten Anfrageergebnissen. Damit lassen sich z. B. inkrementelle Anfragen realisieren. Der Anfragedienst übernimmt die Vor- und die Nachverarbeitung von Anfragen. Während bei der Vorverarbeitung Nutzeranfragen erweitert werden, indem in diese Daten der Nutzerprofile (Nutzerinteressen) eingeflochten werden, werden bei der Anfragenachbearbeitung berechnete Anfrageergebnis nachträglich behandelt. Dabei kann etwa eine Sortierung vorgenommen oder die Ergebnismenge eingeschränkt werden, so daß das nachbearbeitete Anfrageergebnis den Anforderungen des Nutzers entspricht.

Profildienst Alle nutzerbezogenen Daten innerhalb der Personalisierungskomponente werden vom Profildienst verwaltet und den verschiedenen Diensten bereitgestellt. Zu diesen Daten zählen im Wesentlichen Nutzerinteressen, die mit Hilfe von Metadaten in verbreiteten Formaten (Dublin Core, Majour) beschrieben werden. Aber auch Einstellungen, die zur automatischen Nachbearbeitung von Anfrageergebnissen benötigt werden, werden vom Profildienst verwaltet.

Erweiterter Lokaler Dokumentenserver Der in [Tit99] entwickelte Lokale Dokumentenserver wurde hier um die Fähigkeit erweitert, Dokumente nutzergesteuert zu verwalten. Dies bedeutet, daß Nutzer mit diesem Dokumentenserver eigene Dokumente speichern oder allgemein verwalten und anderen Nutzern zugänglich machen können. Dieser Server wird in den Benachrichtigungsprozeß eingebunden, so daß andere Nutzer über Änderungen an den enthaltenen Dokumenten informiert werden können.

Anbindung klassischer Bibliotheksdienste Neben den typischen Diensten einer Digitalen Bibliothek werden auch die Dienste einer klassischen Bibliothek angebunden, da diese auch im “Zeitalter Digitaler Bibliotheken” benötigt werden. Zu diesen klassischen Bibliotheksdiensten zählen etwa die Katalogsuche oder das Abwickeln einer Fernleihe.

Annotationsdienst Der Annotationsdienst gibt dem Nutzer die Möglichkeit, Bemerkungen über Dokumente der Digitalen Bibliothek zu verfassen. Diese Bemerkungen können einerseits mit anderen Nutzern ausgetauscht und andererseits durch die Vergabe von Zugriffsrechten vor dem Zugriff bestimmter Nutzer geschützt werden.

Wie in Kapitel 5 beschrieben, wurden ein Profil- und ein Benachrichtigungsdienst prototypisch implementiert. Der hier erstellte Benachrichtigungsdienst wurde an einen externen Alerting-Server gekoppelt. Kern der Implementierung war jedoch nicht nur eine einfache Koppelung, sondern gleichzeitig eine Entlastung dieses externen Servers. Diese Entlastung basiert auf der Zusammenfassung von Profilen nach dem in Abschnitt 4.2.1.3 vorgestellten und innerhalb dieser Arbeit entwickelten Algorithmus zur Generalisierung von Nutzerprofilen. Die Konfiguration des Benachrichtigungsdienstes erfolgt über Nutzerprofile, welche mittels des Profildienstes verwaltet werden.

Um einen Zugang zu den implementierten Diensten bereitzustellen, wurde außerdem eine Web-Schnittstelle implementiert, über die der Nutzer sein Interessenprofile editieren kann. Diese Art der Schnittstelle bietet den großen Vorteil, daß keine besondere Software

notwendig ist, um auf das System zuzugreifen, lediglich ein Web-Browser, der heute wohl auf fast jedem Rechner installiert ist.

6.2 Ausblick

Da die Themenpalette der Personalisierung und der Digitalen Bibliotheken sehr umfangreich ist, ergeben sich auf der Basis dieser Arbeit eine Vielzahl von Möglichkeiten zur Erweiterung und Verbesserung der hier vorgestellten Dienste.

Eine Rolle entscheidende Rolle könnte dabei das Relevance Feedback einnehmen. So wäre es beispielsweise denkbar, die Konfiguration des Benachrichtigungsdienste nicht allein durch Editieren von Interessenprofilen durch den Nutzer durchzuführen, sondern mit einem Mechanismus, der es dem Nutzer erlaubt, die Güte zu bewerten, die die Alerting-Daten seiner Meinung nach bezüglich seiner Profile haben. Daneben besteht außerdem die Möglichkeit, die Konfiguration von Profilen anhand von Beispieldokumenten vorzunehmen, wie etwa in [Yoc94] oder [BSAS94] beschrieben. Dabei gibt der Nutzer dem System Dokumente (Trainingsmenge) vor, die beispielhaft für seine Interessen sind. Das System analysiert diese Dokumente und erstellt daraus selbstständig ein Interessenprofil.

Weiterhin ist der Einsatz von Techniken aus dem Information Retrieval an verschiedenen Stellen wünschenswert. Beispielsweise könnten Profile vor dem Speichern bereinigt werden, indem die enthaltenen Begriffe auf ihre Stammform reduziert (Stemming) und Stopwörter eliminiert würden. Beim Filtern von Alerting-Daten gegen Profile könnte die Verwendung von Stemming und Stoppwortlisten zu besseren Ergebnissen führen, als ein einfacher Test auf Substrings. Die Nutzung eines speziellen Volltext-Index (Ngram) würde beim Profile-Filtering eine gewisse Toleranz ermöglichen, die dazu führt, daß Eingabefehler des Nutzers beim Editieren der Profile bis zu einem gewissen Grad ignoriert werden könnten.

Ein weiterer Punkt beinhaltet den Ausbau des Ähnlichkeitsmaßes von Profilen, der z. T. auf einem Substring-Kriterium aufbaut. Dieses Kriterium könnte etwa durch den Einsatz von Synonymen und der Durchführung von Stammwortreduktionen ersetzt werden. Ein grundsätzliche Erweiterung der Digitalen Bibliothek besteht in der Aufhebung der Restriktion, ausschließlich Dokumente zu verwalten. Darum besteht eines der Ziele darin, beliebige multimediale Ressourcen zu betrachten.

Profilverwaltung

A.1 Beispiel zur Generalisierung von Profilen

Gegeben seien die folgenden Profile P_1 bis P_5.

P_1

Attribut	Typ	Wert
a_1	Creator	Andreas Heuer
a_2	Title	Datenbanken
a_3	Publisher	Addison Wesley
a_4	Date	1997
a_5	Classification	H.2.8

P_2

Attribut	Typ	Wert
b_1	Creator	Heuer
b_2	Title	OO Datenbanken
b_3	Title	Datenbanken
b_4	Date	97
b_5	Contributor	Schlegelmilch
b_6	Classification	H.2

P_3

Attribut	Typ	Wert
c_1	Creator	Uhrmacher
c_2	Title	Simulation
c_3	Title	Modeling
c_4	Date	2000

P_4

Attribut	Typ	Wert
d_1	Creator	Holger Meyer
d_2	Creator	Meike Klettke
d_3	Title	XML
d_4	Title	Datenbanken

P_5

Attribut	Typ	Wert
e_1	Creator	Andreas Heuer
e_2	Creator	Holger Meyer
e_3	Creator	Meike Klettke
e_4	Title	Datenbanken

Beispiel A.1 Zu berechnen sind die generalisierten Profile G_1 bis G_i. Der zum Generalisieren benötigte Schwellwert k beträgt 0.5.

Im ersten Schritt ergibt sich das erste allgemeine Profil aus dem ersten konkreten Profil: $G_1 := P_1$:

G_1

Attribut	Typ	Wert
g_1	Creator	Andreas Heuer
g_2	Title	Datenbanken
g_3	Publisher	Addison Wesley
g_4	Date	1997
g_5	Classification	H.2.8

Im nächsten Schritt wird das Profil P_2 generalisiert. Da die Attribute g_2 und b_3 identisch sind, für die Attributpaare $(g_1,b_1),(g_2,b_2),(g_4,b_4)$ das Substringkriterium gilt und die Ähnlichkeit $sim_C(g_5,b_6) = \frac{2}{3}$ beträgt und somit größer k ist, ergibt sich die Ähnlichkeit zwischen G_1 und P_2 wie folgt:

$$sim_P(G_1,P_2) = \frac{1+0.5\cdot 3+\frac{2}{3}}{6} = \frac{19}{36} \geq k$$

Da die Ähnlichkeit größer als der Schwellwert k ist, wird P_2 nun G_1 hinzugefügt. Dabei werden g_1 durch b_1, g_4 durch b_4 und g_5 durch b_6 ersetzt, g_3 wird entfernt. Das neue G_1 ergibt sich damit aus

G_1

Attribut	Typ	Wert
g_1	Creator	Heuer
g_2	Title	Datenbanken
g_4	Date	97
g_5	Classification	H.2

Die Ähnlichkeit der Profile G_1 und P_3 beträgt 0, da weder identische noch ähnliche Werte zwischen beiden vorliegen. Somit ergibt sich ein neues generalisiertes aus dem konkreten Profil P_3: $G_2 := P_3$.

G_2

Attribut	Typ	Wert
h_1	Creator	Uhrmacher
h_2	Title	Simulation
h_3	Title	Modeling
h_4	Date	2000

Zwischen den Profilen P_4 und G_1 besteht eine Ähnlichkeit von $\frac{1}{4}$, die Ähnlichkeit zwischen P_4 und G_2 ist 0, d. h. der Schwellwert von $k = 0.5$ wird nicht erreicht. Demzufolge wird das Profil P_4 in die Menge der generalisierten Profile übernommen: $G_3 := P_4$.

G_3		
Attribut	Typ	Wert
l_1	Creator	Holger Meyer
l_2	Creator	Meike Klettke
l_3	Title	XML
l_4	Title	Datenbanken

Um schließlich das Profil P_5 zu generalisieren, werden die Ähnlichkeiten zwischen P_5 und allen G_i berechnet:

$$\begin{aligned} sim_P(G_1, P_5) &= \frac{1+0.5\cdot 1+0}{5} = \frac{3}{10} < k \\ sim_P(G_2, P_5) &= \frac{0+0+0}{5} = 0 < k \\ sim_P(G_3, P_5) &= \frac{2+0.5\cdot 2+0}{5} = \frac{3}{5} \geq k \end{aligned}$$

Da nur die Ähnlichkeit von P_5 zu G_3 ausreichend groß ist, wird nun P_5 durch Generalisierung G_3 hinzugefügt. Die Attribute e_4 und l_3 bzw. e_5 und l_4 sind identisch, e_2 ist Substring von l_1 und e_3 ist Substring von l_2. Das Profil G_3 ändert sich nach Hinzufügen von P_5 zu

G_3		
Attribut	Typ	Wert
l_1	Creator	Meyer
l_2	Creator	Klettke
l_3	Title	XML
l_4	Title	Datenbanken

A.2 Profile-Filtering

Ändern sich Dokumentenbestände, etwa durch Hinzufügen eines neuen Konferenzbandes, so werden Digitale Bibliotheken darüber informiert. Diese Bibliotheken benachrichtigen daraufhin den Teil ihrer Benutzer, der sich für die Dokumente interessiert, die im Bestand geändert, in diesem Fall hinzugefügt wurden. Um herauszufinden, welche Nutzer sich nun für welche Änderungen des Bestandes interessieren, müssen die Änderungen mit den Nutzerprofilen verglichen werden, in denen die Interessen der einzelnen Nutzer kodiert sind. Der Vergleich zwischen Profilen und Änderungen, die in Form von Metadaten beschrieben werden bzw. das Überprüfen, ob ein Profil auf ein Dokument und seine Metadaten paßt, wird als Profilfilterung (Profile-Filtering) bezeichnet.

A.2.1 Grundsätzliches

Ein Profil besteht aus einer Menge von Begriffen, wobei ein Begriff eine beliebige Zeichenkette sein und damit auch aus mehreren Wörtern bestehen kann. Der Interpretation von Profilen liegt nun zugrunde, daß *alle* Wörter eines Profils in dem Objekt (Dokument,

Metadaten) in identischer Form oder als Substring davon auftreten müssen, damit dieses Profil als auf das Objekt passend bezeichnet werden kann. Die Begriffe eines Profils sind also konjunktiv miteinander verknüpft. Um Begriffe disjunktiv miteinander zu verknüpfen, müssen diese jeweils in verschiedene Profile aufgeteilt werden. Negationen sind grundsätzlich möglich, sollen hier aber nicht betrachtet werden.

Die folgenden Abschnitte geben eine kurze Übersicht über verschiedene Verfahren, die zum Filtern von Metadaten gegen Nutzerprofile verwendet werden können. Für eine Analyse weiterer Verfahren sei hier auf [FD92] verwiesen.

A.2.2 Vektorraummodell

Das Vektorraummodell stellt eine Möglichkeit dar, um zu überprüfen, ob ein Dokument relevant zu einem gegebenen Profil ist (siehe etwa [YGM93a]). Dazu werden das Profil und das zu untersuchende Dokument jeweils als Vektor betrachtet. Ein solcher Vektor besteht aus Paaren (r_i, w_i), wobei w_i ein Wort ist und r_i ein Ranking-Wert, der die Häufigkeit des Auftretens von w_i innerhalb eines Dokumentes bzw. eines Profils beschreibt. Tritt ein Wort w_i innerhalb eines Dokumentes nicht auf, so beträgt der Ranking-Wert $r_i = 0$. Wichtig ist, daß solche Vektoren alle möglichen Wörter enthalten, die in zu analysierenden Dokumenten oder Profilen auftreten können, mit Ausnahme von Stopwörtern und gebeugten Formen. D. h. in den Vektoren exisitieren nur Wörter in ihrer Stammform, die keine Stopwörter sind. Außerdem müssen die Wörter immer an derselben Stelle der Vektoren stehen.

Um nun zu überprüfen, wie relevant ein Dokument verglichen mit einem Profil ist, wird zum einen der Vektor $P = ((r_1, s_1) \ldots, (r_n, s_n))$ mit r_i Ranking-Wert und s_i Wort, für das Profil aufgebaut, und zum anderen der Vektor $D = ((t_1, u_1), \ldots, (t_n, u_n))$ für das Dokument, mit t_i Ranking-Wert und u_i Wort. Im Anschluß daran werden die Wörter in den Vektoren ausgeblendet, so daß nur noch die Ranking-Werte enthalten sind. Die Relevanz bzw. Ähnlichkeit $sim(D,P)$ dieser Vektoren ergibt sich dann aus

$$sim(D,P) = \frac{D \cdot P}{\|D\| \cdot \|P\|} = \frac{\sum_{i=1}^{n} r_i \cdot t_i}{\sqrt{\sum_{i=1}^{n} r_i^2 \sum_{i=1}^{n} t_i^2}}$$

Sind die Vektoren normalisiert, haben sie also einen Betrag von 1, dann ergibt sich die Ähnlichkeit aus

$$sim(D,P) = D \cdot P = \sum_{i=1}^{n} r_i \cdot t_i$$

Dieses Modell ist für das Filtern von Dokumenten gegen Profile in Digitalen Bibliotheken eher ungeeignet, da es im Wesentlichen Profile findet, die dem Dokument ähnlich sind. Gesucht werden aber Profile, die vollkommen darauf passen. Dies wäre zwar unter Verwendung der Vektorraummodells möglich, indem eine Ähnlichkeit von 1 gefordert wird. Da dies aber die Identität der aufgebauten Vektoren bedeutet, ist etwa die Berechnung des Vektorproduktes überflüssig, ein komponentenweiser Vergleich der Vektorelemente wäre ausreichend. Darauf zielt das Verfahren jedoch nicht ab.

A.2.3 Bool'sches Modell

In [YGM93b] werden Verfahren und Indexstrukturen vorgestellt, mit deren Hilfe Volltextdokumente gegen Profile gefiltert werden können. Die Grundlage dieser Verfahren ist das Bool'sche Retrieval-Modell, daß aus dem klassischen Information Retrieval stammt. In diesem Modell muß ein Dokument alle Anfrageforderungen erfüllen, um in die Ergebnismenge aufgenommen zu werden. Bezogen auf das Profile-Filtering bedeutet dies, daß ein Dokument bzw. seine Metadaten alle Begriffe eines Profils enthalten muß, um zu diesem Profil zu passen.

In den nächsten Abschnitten werden einige der in [YGM93b] vorgestellten Verfahren skizziert, mit deren Hilfe Dokumente, aufbauend auf dem bool'schen Retrieval-Modell, gegen Profile geparst werden können. Auf eine detailierte Beschreibung der Optimierungen, mit denen sich die Effizienz dieser Verfahren noch deutlich steigern läßt, wird hier jedoch verzichtet.

A.2.3.1 Naive Methode – Brute Force

Grundlage der naiven Methode (und auch der beiden nächsten Methoden) ist eine Analyse der Dokumente, die gegen Profile geparst werden. Bei dieser Analyse wird eine Menge aufgebaut, die alle im Dokument auftretenden Wörter beinhaltet. Im Anschluß daran werden alle Profile daraufhin überprüft, ob die Wörter, die sich im Profil befinden, auch in der Menge der Wörter des Dokumentes vorkommen. Ist dies der Fall, so zählt das jeweilige Profil als zum Dokument passend.

A.2.3.2 Zählmethode

In der Zählmethode wird, im Gegensatz zur naiven Methode, ein Index benutzt, der den Zugriff auf die Profile beschleunigt. Dieser Index besteht aus einer invertierten Liste, in der die Wörter aller Profile eingetragen sind. Dabei werden jedem Wort diejenigen Profile zugeordnet, in denen dieses Wort vorkommt. Ein Profil mit k Worten ist somit in k verschiedenen Indexeinträgen zu finden. Zusätzlich zu diesem Index werden zwei Felder (Arrays) aufgebaut, von denen das erste (TOTAL) die Anzahl der Wörter der Profile speichert. Das zweite (COUNT) zählt, in wievielen Indexeinträgen ein Profil vorkommt. Stimmen der Werte für COUNT und TOTAL eines Profils überein, dann gilt das Profil als zum zu untersuchenden Dokument passend.

Beispiel A.2 In Abbildung A.1 ist ein Beispiel angegeben, an dem die Zählmethode demonstriert werden soll. Das zu untersuchende Dokument soll die Wörter a, b, c und f enthalten. Aus der invertierten Liste wird ersichtlich, daß beispielsweise das Wort a im Profil P_1, P_2 und P_3 vorkommt, das Wort c nur in Profil P_5. Das Array TOTAL zeigt, daß etwa die Profile P_1, P_2 und P_4 aus jeweils zwei Wörtern bestehen. Die Analyse gschieht nun wie folgt:

1. Alle Werte in COUNT werden mit 0 initialisert.

2. Alle Wörter aus dem Dokument werden in die Wortmenge aufgenommen.

3. Für jedes Wort aus der Wortmenge wird mit Hilfe des Index überprüft, in welchen Profilen es auftritt. Für die Profile, in denen dieses Wort vorkommt, wird der Wert in COUNT inkrementiert. Da etwa das Wort *a* in den Profilen P_1, P_2 und P_3 vorkommt, wird deren COUNT-Wert inkrementiert.

4. Weil nach Durchführung dieser Methode die Werte von COUNT und TOTAL jeweils für die Profile P_1 (2) und P_4 (2) identisch sind, passen diese auf das Beispieldokument.

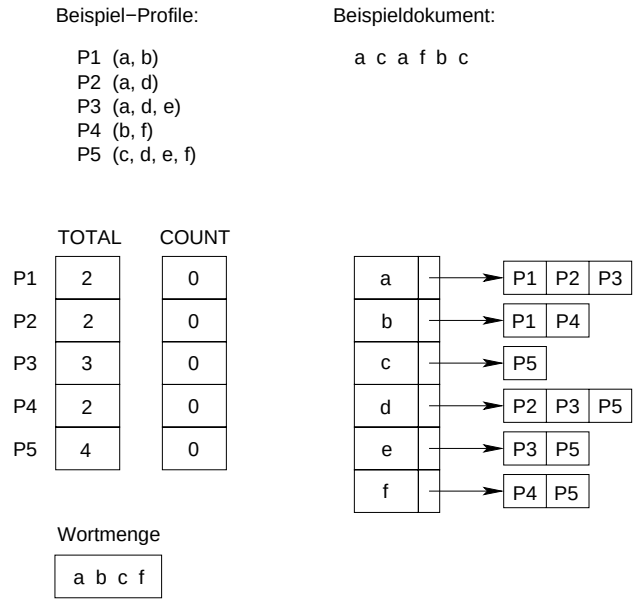

Abbildung A.1: Beispiel zur Zählmethode

A.2.3.3 Schlüsselmethode

Grundlage der Schlüsselmethode ist wieder eine invertierte Liste, die den verschiedenen Wörtern zuordnet, zu welchen Profilen sie gehören. Diese Liste enthält zum einen die Menge der Wörter, hier auch Schlüssel genannt, die in den verschiedenen Profilen vorkommen. Diese Wörter sind jeweils mit einer Liste verknüpft, aus der zum einen hervorgeht, in welchen Profilen diese Wörter auftreten. Zum anderen ist verzeichnet, aus wievielen Wörtern diese Profile abzüglich des dort verzeichneten bestehen. Diese Wörter sind im Anschluß an die Anzahl der vertretenen Wörter vermerkt. Durch diesen Aufbau erscheinen die Wörter eines Profils nur in einem Eintrag der invertierten Liste, und nicht in *k* verschiedenen, wie bei der Zählmethode. Die Analyse eines Dokumentes soll am folgenden Beispiel vorgeführt werden.

Beispiel A.3 In Abbildung A.2 sind wieder ein Beispieldokument und eine Menge von Profilen angegeben. Aus den Profilen wird die invertierte Liste wie folgt aufgebaut: Das Wort *a* kommt in den Profilen P_1, P_2 und P_3 vor. Dabei enthält das Profil P_1 außerdem mit

b noch ein weiteres Wort, P_3 enthält außer *a* noch die beiden Wörter *d* und *e*. Der weitere Aufbau erfolgt analog.

Um zu analysieren, welche der Profile auf das Dokument passen, werden zunächst die Wörter des Dokumentes in die Wortmenge aufgenommen. Danach wird für jedes dieser Wörter nachgesehen, ob es laut invertierter Liste in einem Profil vorkommt. Ist dies der Fall, dann wird überprüft, ob die Wörter, die sich außerdem im Profil befinden, auch im Dokument auftauchen. Dazu werden die Wörter, die in der invertierten Liste hinter dem Profilnamen befinden, in der Wortmenge gesucht. Verläuft diese Suche erfolgreich, dann paßt das Profil auf das Dokument.

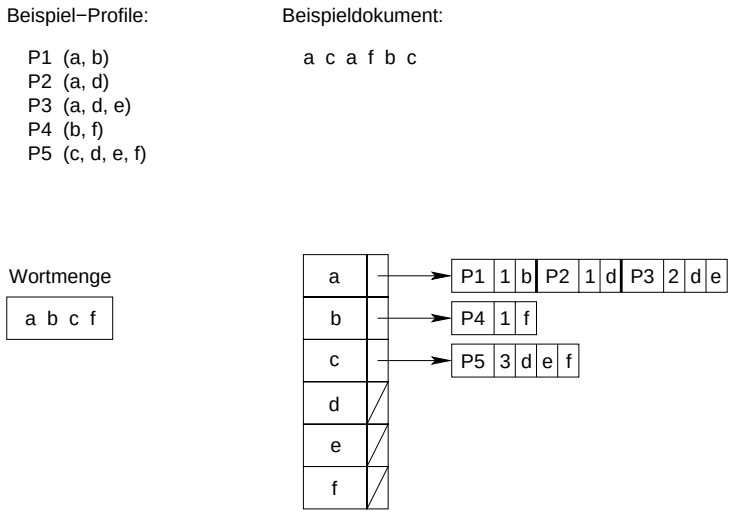

Abbildung A.2: Beispiel zur Schlüsselmethode

A.2.3.4 Bewertung

Die auf dem bool'schen Modell basierenden Verfahren finden genau die Profile, die vollständig auf ein gegebenes Dokument passen. Da sie aber nur für komplette Wörter eines Dokumentes bzw. eines Profils funktionieren und eine Substring-Suche ausschließen, sind sie in dieser Form für die Verwendung in Digitalen Bibliotheken ungeeignet.

A.2.4 Verwendung endlicher Automaten

In [Fau01] wird ein Verfahren zum effizienten Profile-Filtering vorgestellt, daß auf der Benutzung endlicher Automaten beruht. Die dabei verwendeten Automaten sind sogenannte GSMs (Deterministic Generalized Sequential Machine). Das Verfahren funktioniert nun wie folgt:

Zuerst wird anhand der Profile ein GSM konstruiert. Dieser besteht aus Zuständen und Zustandsübergängen. Ein Zustandsübergang ist gekennzeichnet durch einen Buchstaben aus dem Alphabet, aus dem alle Wörter der Dokumente und aller Profile gebildet werden. Die Eingabe des GSMs ist ein Wort des zu untersuchenden Dokumentes. Dieses

Wort wird Buchstabe für Buchstabe eingelesen, wobei jeder Buchstabe einen Zustandsübergang im Automaten bedeutet (die Analyse beginnt im Startzustand des Automaten). Ist das Wort in einem der Profile enthalten bzw. ist das Wort eines Profils Substring dieses Wortes, dann gibt der Automat einen Index aus, der das entsprechende Profil identifiziert. Wurde das Wort vollständig analysiert, so besteht die Ausgabe des Automaten aus einer Indexmenge, wobei die Elemente dieser Menge jeweils die Profile identifizieren, die darauf passen. Das folgende Beispiel soll diese knappe Erläuterung illustrieren, eine ausführliche Schilderung kann [Fau01] entnommen werden.

Beispiel A.4 Gegeben sei eine Menge W einfacher Profile, die aus lediglich einem Wort bestehen, mit $W = \{w_1, w_2\} \subseteq \{a,b,c,d\}^*$, wobei $w_1 =$ abc und $w_2 =$ bcd. Der dazugehörige GSM ist in Abbildung A.3 dargestellt. Dabei ist q_0 Anfangszustand, alle Zustände sind auch Endzustände. Ein Übergang c kennzeichnet dabei die Transition, die durch Einlesen des Buchstaben c ensteht, beim Übergang $c|1$ wird zusätzlich eine 1 ausgegeben, die kenzeichnet, das w_1 im bisher bearbeiteten String enthalten ist, oder daß dieser String mit w_1 identisch ist.

Soll nun ein Dokument D analysiert werden, welches lediglich das Wort *abcdacdab* enthält, so führt dies über die Zustände q_0, q_1, q_2, q_4, q_0 u. s. w., wobei die Indizies 1 und 2 ausgegeben werden. Das bedeutet, daß beide Profile w_1 und w_2 auf das Dokument D passen.

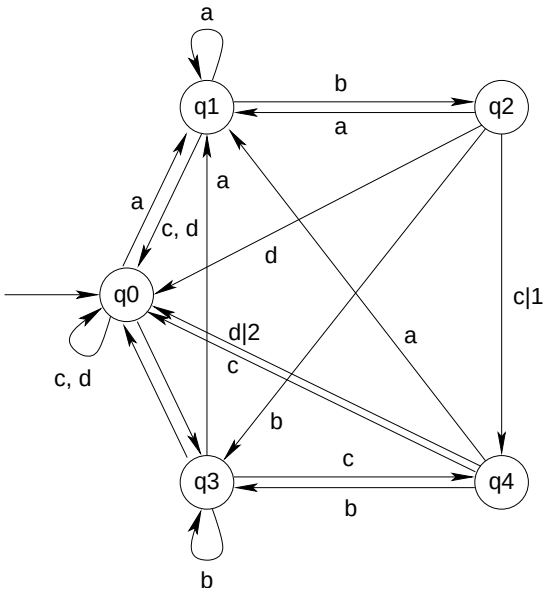

Abbildung A.3: Ein Beispiel-GSM

Das hier skizzierte, auf endlichen Automaten beruhende Verfahren findet die Profile, die genau auf gegebene Dokumente passen, d. h. deren Wörter im Dokument auftreten oder Substrings der Wörter im Dokument sind, was als großer Vorteil insbesondere im Vergleich zu den o. g. Verfahren zu werten ist. In [Fau01] wird außerdem gezeigt, daß die

Analyse mit den GSMs effizient ist. Ein Nachteil ergibt sich jedoch aus der Tatsache, daß bei jeder Änderungen der Profile die dazugehörigen Automaten angepaßt werden müssen.

A.3 Import/Export von Profilen

Die nachfolgend abgebildete DTD (Document Type Definition) beschriebt eine Struktur, in der Profildaten mittels XML dargestellt werden können. Diese DTD beschreibt jedoch nur einen kleinen Teil der Daten, die sich innerhalb eines Profils speichern lassen. In Abschnitt A.3.2 wird ein Beispiel-Dokument gezeigt, das dieser DTD entspricht.

A.3.1 DTD

```
<!ELEMENT Profiles (Profile*)>
<!ELEMENT Profile (User, Interests, Property*)>


<!ELEMENT User (Username, Fullname, EMail)>
<!ELEMENT Username (#PCDATA)>
<!ELEMENT Fullname (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>


<!ELEMENT Interests (Interest+)>
<!ELEMENT Interest (#PCDATA)>
<!ATTLIST Interest
  Type (Author|Title|Abstract|AbstractTitle|Journal) #REQUIRED>

<!ATTLIST Interest
  NotificationPolicy (daily|weekly|monthly) #REQUIRED>

<!ELEMENT Property (ResultProperty?)>
<!ELEMENT ResultProperty (MaxResultTuples?)>
<!ELEMENT MaxResultTuples (#PCDATA)>
```

A.3.2 Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE Profiles SYSTEM "Profile.dtd">

<Profiles>
  <Profile>
    <User>
      <Username>zeitz</Username>
      <Fullname>Andre Zeitz</Fullname>
      <Email>zeitz@informatik.uni-rostock.de</Email>
    </User>
    <Interests>
      <Interest NotificationPolicy="weekly"
       Type="abstracttitle">database</Interest>
      <Interest NotificationPolicy="daily"
```

```
      Type="title">digital library</Interest>
    </Interests>
    <Property>
      <ResultProperty>
        <MaxResultTuples>30</MaxResultTuples>
      </ResultProperty>
    </Property>
  </Profile>
</Profiles>
```

A.4 Beispiel einer Hermes-Mail

```
<?xml version="1.0" encoding="UTF-8"?>
<notification>
<profilename>62bkrdxrazn5prkxig5amiolt</profilename>
<profileid>51282</profileid>
<document>
<messageid>52111</messageid>
<score>0.00</score>
<header>
<issue>
<pinfo>
<pnm>Springer-Verlag</pnm>
<loc>Berlin/Heidelberg</loc>
</pinfo>
<jinfo>
<jid>00216</jid>
<jtl>Fresenius' Journal of Analytical Chemistry</jtl>
<jabt>Fresenius J Anal Chem</jabt>
<issn>0937-0633</issn>
</jinfo>
<pubinfo>
<vid>371</vid>
<iid>4</iid>
<cd year="2001" month="10" type="print">
<cd year="2001" month="8" type="online" day="31">
<doi>10.1007/s002160100925</doi>
</cd>
</cd>
</pubinfo>
</issue>
<artcon>
<genhdr language="EN">
<artinfo>
<aid>00925</aid>
<artty artty="RP">
<categ>Special Issue Paper</categ>
<ppf>467</ppf>
<ppl>473</ppl>
<crn>Springer-Verlag 2001</crn>
<hst>
```

```
<re day="02" month="04" year="2001">
<rv day="23" month="04" year="2001">
<acc day="08" month="05" year="2001">
<misc day="31" month="8" year="2001">
Electronic Publication</misc>
</acc>
</rv>
</re>
</hst>
</artty>
</artinfo>
<tig>
<atl>Characterization of wastewater toxicity by means of a
whole-cell bacterial biosensor, using <e5>Pseudomonas putida</e5>,
in conjunction with chemical analysis</atl>
</tig>
<aug>
<au>
<fnms>Marinella</fnms> <snm>Farr??</snm>
<orf rid="A1"/>
</au>
<au>
<fnms>Dami? </fnms> <snm>Barcel??</snm>
<orf rid="A1"/>
</au>
<aff>
<oid id="A1">
<oad>Department of Environmental Chemistry IIQAB-CSIC,
Jordi Girona 18-26, 08034 Barcelona, Spain</oad>
</oid>
</aff>
</aug>
<abs>
<p>
<e6>Abstract.</e6>A new amperometric biosensor based on
inhibition of<e5>Pseudomonas putida</e5>has been developed
to assess the acute toxicity of wastewater. This system
uses the biological component immobilized on disposable
screen-printed electrodes. The responses for a selected
group of polar organic standard substances were studied
using<e5>Pseudomonas putida</e5>as biological component.
The results have been compared with responses
obtained using the same system and<e5>Escherichia coli</e5>
as biological component and with the bioluminescence
inhibition of<e5>Vibrio fisheri</e5>using ToxAlert 100.
Different properties, e.g. the standard deviation (SD) of
the data, the goodness of fit (R<sup>2</sup>) and the
standard deviation (Syx) of the vertical distances of the
points from the inhibition curve, the 50% effective
concentration (EC<inf>50</inf>) and the toxicity units
(TII<inf>50</inf>) of the standard substance, were
calculated and compared. This biosensor was used to
```

```
assess the acute toxicity of real wastewater samples
collected at different wastewater treatment plants
(WWTP).</p><p>Finally, a sequential solid-phase
extraction (SSPE) procedure followed
by liquid chromatographya??mass spectrometry (LCa??MS)
was used to determine the polar organic toxic substances
present in the wastewater samples.</p>
</abs>
<aloc contenttype="text/html"
href="http://link.springer.de/link/service/journals/
00216/bibs/1371004/13710467.htm"/>
</genhdr>
</artcon>
</header>
</document>

...

</notification>
```

Abbildungen

Literatur

[ACM98] *The ACM Computing Classification System*. Technischer Bericht, Association of Computer Machinery (ACM), 1998. *http://www.acm.org/class.*

[BPSMM00] BRAY, TIM, JEAN PAOLI, C. M. SPERBERG-MCQUEEN und EVE MALER (Herausgeber): *Extensible Markup Language (XML), Version 1.0 (Second Edition)*. World Wide Web Consortium, Oktober 2000. W3C Recommendation, verfügbar unter: *http://www.w3.org/TR/2000/REC-xml-20001006.pdf*.

[BSAS94] BUCKLEY, CHRIS, GERARD SALTON, LAMES ALLAN und AMIT SINGHAL: *Automatic Query Expansion Using SMART*. In: *Proceedings of the Third Text REtrieval Conference (TREC-3), Gaithersburg, Maryland*, 1994.

[BYRN99] BAEZA-YATES, RICARDO und BERTHIER RIBEIRO-NETO: *Modern Information Retrieval*. Addison Wesley Longman, 1. Auflage, 1999.

[cor98] *CORBA/IIOP Specification*, 1998. *http://www.omg.org.*

[Cow01] COWARD, DANNY: *Java Servlet Specification*. Sun Microsystems, 2001.

[CW99] CAMPIONE, MARY und CATHY WALRATH: *The Java Tutorial*. Addison-Wesley, 2. Auflage, 1999.

[DC99] *Dublin Core Metadata Element Set, Version 1.1: Reference Description*. Technischer Bericht, Juli 1999.

[DDC] *Dewey Decimal Classification System*. Technischer Bericht, Online Computer Library Center (OCLC). *http://www.olcl.org/fp/about/index.htm.*

[DDV+01] DATTA, ANINDYA, KAUSHIK DUTTA, DEBRA VANDERMEER, KRITHI RAMAMRITHAM und SHAMKANT B. NAVATHE: *An Architecture to Support Online Personalization on the Web*. The VLDB Journal, 10(1):104–117, April 2001.

[EF00] ENDRES, ALBERT und DIETER W. FELLNER: *Digitale Bibliotheken: Informatik Lösungen für globale Wissensmärkte*. dpunkt.Verlag, 1. Auflage, 2000.

[EHS+00] ENDIG, MARTIN, MICHAEL HÖDING, GUNTER SAAKE, KAI-UWE SATTLER und EIKE SCHALLEHN: *Federation Services for Heterogeneous Digital Libraries Accessing Cooperative and Non-cooperative Sources*. In: *Kyoto International Conference on Digital Libraries: Research and Practice*, Seiten 314–321, Kyoto University, Kyoto, Japan, November 2000.

[Fau01] FAULSTICH, LUKAS: *Efficient Multi-Profile Filtering using Finite Automata*. Technischer Bericht, Institut für Informatik, Freie Universität Berlin, April 2001.

[FD92] FOLTZ, PETER W. und SUSAN T. DUMAIS: *Personalized Information Delivery: An Analysis of Information Filtering Methods*. Communcations of the ACM, 35(12):51–60, Dezember 1992.

[FFS+01] FAENSEN, D., L. FAULSTICH, H. SCHWEPPE, A. HINZE und A. STEIDINGER: *HERMES – A Notification Service for Digital Libraries*. In: *Proceedings of the First ACM/IEEE Joint Conference on Digital Libraries (JCDL), Roanoke, VA, USA*, Seiten 373–380, Juni 2001.

[GHMN01] GUDGIN, MARTIN, MARC HADLEY, JEAN-JACQUES MOREAU und HENRIK FRYSTYK NIELSEN (Herausgeber): *SOAP Version 1.2, Working Draft*. World Wide Web Consortium, Oktober 2001. *http://www.w3.org/TR/soap12-part1/*.

[Har00] HAROLD, ELLIOTTE RUSTY: *Java Network Programming*. O'Reilly, 2. Auflage, August 2000.

[HD01] HAASE, PETER und MICHAEL DREWS: *Komplexe Softwaresysteme: Digitale Biblioteken – Ein Virtueller Dokumentenserver*. Universität Rostock, Fachbereich Informatik, 2001.

[HK00] HAN, JIAWEI und MICHELINE KAMBER: *Data Mining: Concepts and Techniques*. Morgan Kaufman, August 2000.

[HMPT00] HEUER, ANDREAS, HOLGER MEYER, BEATE PORST und PATRICK TITZLER: *BlueView: Virtual Document Servers for Digital Libraries*. In: *Proceedings of IEEE Advances in Digital Libraries 2000 (ADL 2000), Washington, DC*, Seiten 207–217, 2000.

[HP99] HEUER, ANDREAS und DENNY PRIEBE: *IRQL – Yet Another Language for Querying Semi-Structured Data?* Preprint, Universität Rostock, Fachbereich Informatik, 1999.

[ing] *CA Open Ingres, Programmer's Guide*. Computer Associates.

[Jon99] JONAS, MICHAEL: *Ein Replikationsdienst für komplexe Dokumente*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999.

[KKPS01] KAHAN, JOSÉ, MARJA-RIITA KOIVUNEN, ERIC PRUD'HOMMEAUX und RALPH R. SWICK: *Annotea: An Open RDF Infrastructure for Shared Web Annotations*. In: *Proceedings of the 10th WWW International Conference, Honkong*, Mai 2001.

[LJD01] LAURENT, SIMON ST., JOE JOHNSTEN und EDD DUMBILL: *Programming Web Services with XML-RPC*. O'Reilly, 1. Auflage, Juni 2001.

[Llo01] LLOPART, EDUARDO PELEGRÍ: *Java Server Pages Specification*. Sun Microsystems, August 2001.

[Maj00] *Majour Header DTD Documentation*. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[Mey00] MEYER, HOLGER: *Datenbank-Anwendungsprogrammierung*. Vorlesungsskript, Oktober 2000.
http://wwwdb.informatik.uni-rostock.de/∼hme/lehre/.

[Mod01] MODI, TARAK: *Clean up your Wire Protocol with SOAP*. Java World, März 2001.

[MSC] *Mathematics Subject Classification*. American Mathematics Society (AMS). *http://www.ams.org/msc.*

[Pae96] PAEPCKE, ANDREAS: *Digital Libraries: Searching Is Not Enough — What We Learned On Site*. D-Lib Magazine, Mai 1996. *http://www.dlib.org/dlib/may96/stanford/05paepcke.html.*

[Riv92] RIVEST, R.: *The MD5 Message-Digest Algorithm*. Technischer Bericht, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[RN95] RUSSELL, STUART und PETER NORVIG: *Artifical Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 1995.

[sql00] *SQL Multimedia and Application Packages (SQL/MM), Part 2: Full-Text*, Februar 2000.

[SS00] SAAKE, GUNTER und KAI-UWE SATTLER: *Datenbanken & Java – JDBC, SQLJ und ODMG*. dpunkt.Verlag, 1. Auflage, 2000.

[THK99] TOCHTERMANN, KLAUS, DAVID HICKS und ANDREAS KUSSMAUL: *Support for Customization and Personalization on the Web*. In: BRA, P. DE und JOHN J. LEGGETT (Herausgeber): *Proceedings of Webnet 99 – World Conference on the WWW and Internet*, Seiten 1048–1053, Honolulu, Hawaii, USA, Oktober 1999.

[Tit99] TITZLER, PATRICK: *Entwurf und Implementierung eines Virtuellen Dokumenten-Servers*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999.

[WFC+99] WHITE, SETH, MAYDENE FISHER, RICK CATTELL, GRAHAM HAMILTON und MARK HAPPNER: *JDBC API Tutorial and Reference*. Addison-Wesley, 2. Auflage, Juli 1999.

[WKLW98] WEIBEL, S., J. KUNZE, C. LAGOZE und M. WOLF: *Dublin Core Metadata Element Set, Vesion 1.0: Reference Description*. Technischer Bericht, Dublin Core Metadata Initiative, September 1998.

[YGM93a] YAN, TAK W. und HECTOR GARCIA-MOLINA: *Index Structures for Information Filtering Under the Vector Space Model*. Technischer Bericht, Department of Computer Science, Stanford University, 1993.

[YGM93b] YAN, TAK W. und HECTOR GARCIA-MOLINA: *Index Structures for Selective Dissemination of Information Under the Boolean Model*. Technischer Bericht, Department of Computer Science, Stanford University, Dezember 1993.

[Yoc94] YOCHUM, JULIAN A.: *Research in Automatic Profile Creation and Relevance Ranking with LMDS*. In: *Proceedings of the Third Text REtrieval Conference (TREC-3), Gaithersburg, Maryland*, 1994.

[Zei00] ZEITZ, ANDRE: *DOM/SAX/RAX*. Hauptseminar, Universität Rostock, Fachbereich Informatik, Lehrstuhl Datenbanken und Informationssysteme, Juni 2000. *http://wwwdb.informatik.uni-rostock.de/Lehre/Vorlesungen/hs_ss2000/zeitz_paper.ps.gz*.

[ZZZ+01] ZEITZ, ANDRE, GREGOR ZIMMERMANN, MATHIAS ZARICK, CHRISTIAN ROMBERG und BJÖRN BELOW: *Ein Virtueller Dokumentenserver für Digitale Bibliotheken*. In: HÖPFNER, HAGEN (Herausgeber): *Beitragsband zum Studierenden-Programm bei der 9. Fachtagung "Datenbanken in Büro, Technik und Wissenschaft"*, Seiten 29–31. GI Fachausschuß 2.5, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, 2001.

Selbstständigkeitserklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, 14.11.2001

Andre Zeitz

Thesen

1. Die individuelle Unterstützung der Nutzer Digitaler Bibliotheken kann auf vielfältige Art und Weise geschehen.
2. Zentrale Konzepte bei dieser Unterstützung sind die Benachrichtigung (Alerting) und die Verwaltung von Nutzerinteressen in sogenannten Profilen.
3. Zur Realisierung der Benachrichtigung können externe Benachrichtigungsdienste so angebunden werden, daß ihre Belastung nur geringfügig steigt. Dabei wird ein großer Teil der anfallenden Last vom externen Benachrichtigungsdienst auf das lokale Bibliothekensystem übertragen.
4. Nutzerinteressen können in Nutzeranfragen eingearbeitet werden, so daß das Ergebnis auf den anfragestellenden Nutzer zugeschnitten wird. Unter der Voraussetzung, daß in einer Digitalen Bibliothek vorwiegend Dokumente verwaltet werden, auf denen eine Volltextsuche durchgeführt wird, eignen sich für eine solche Anfragevorverarbeitung insbesondere Volltext-Anfragesprachen wie der IRQL oder SQL/MM Full Text.
5. Eine personalisierte Digitale Bibliothek bietet dem Nutzer auch die Möglichkeit, seine Dokumente zu verwalten. Der Vorteil gegenüber der herkömmlichen Verwaltung im Dateisystem besteht darin, daß der Organisationsaufwand, zu dem etwa die Datensicherung zählt, vom Nutzer an das Bibliothekensystem übertragen wird.
6. Verschiedene Nutzer können mit Hilfe einer Digitalen Bibliothek kooperieren. Die Grundlage dieser Kooperation ist die Vergabe von Zugriffsrechten durch den Nutzer auf einzelne Ressourcen wie etwa Dokumente. Damit wird der Zugriff unberechtigter Nutzer auf diese Ressourcen verhindert.
7. Die von einer klassischen Bibliothek bereitgestellten elektronischen Dienste können an eine Digitale Bibliothek gekoppelt und erweitert werden.
8. Die Implementierung von Personalisierungsdiensten ist auf Basis einer Dienstearchitektur und unter Benutzung einer High-Level-Programmiersprache relativ einfach realisierbar.