Entwicklung eines mobilen Clients für digitale Bibliotheken

Studienarbeit

eingereicht an der	Fakultät für Ingenieurwissenschaften an der Universität Rostock
Autor:	Sebastian Dolke
Matrikel-Nr.:	098201235
Studiengang:	Informatik
Betreuer:	Dr.-Ing. Holger Meyer Dipl. Inf. Birger Hänsel
Lehrstuhl:	Datenbank- und Informationssysteme

Zusammenfassung

Durch die ständig wachsende Menge digitaler Informationen und nicht zuletzt auch durch den Ausbau der globalen Netze haben digitale Bibliotheken in den letzten Jahren ständig an Bedeutung gewonnen. Elektronische Dokumente sind besser aktualisierbar und der Zugang ist unkomplizierter und schneller als auf konventionelle Bücher oder Zeitschriften. Neben leistungsfähigen Systemen zur Bereitstellung digitaler Informationen, werden Applikationen für den Zugriff und die persönliche Verwaltung von elektronischen Dokumenten benötigt.

Aufgabe dieser Studienarbeit war die Entwicklung eines Systems, bestehend aus einem mobilen Client und einem Dokumentenserver, welches Konferenzbesuchern den Zugriff auf die digitalen Inhalte einer Konferenzdomäne ermöglicht. Hierfür werden Anforderungen diskutiert, welche die einzelnen Komponenten eines solchen Systems erfüllen müssen und Konzepte zur Lösung der verschiedenen Probleme vorgestellt. Die Entwicklung einer einsatzfähigen Client-Applikation und der Aufbau eines Dokumentenservers waren das praktische Ziel dieser Arbeit.

Abstract

Because of the steady growing amount of digital informations and also because of the extension of the global networks, digital libraries became more and more important during the last years. The update of electronic documents is easier and faster and they are better accessible as conventional books and articles. Beside capable systems to provide digital informations, we need applications to access and manage electronic documents.

In the scope of this thesis a system that consist of a mobile client and a document server should be developed, that allows conference attendees to access the digital content of a conference domain. Therefore we will discuss the requirements that the components of that kind of system has to match and we will show solutions for the pointed out problems. The development of a ready to use client application and a working document server was the practical goal of this work.

CR-Klassifikation

- H.3.3 Information Search and Retrieval
- H.3.7 Digital Libraries
- H.5.2 User Interfaces

Keywords

Digital Libraries, Mobile Computing, User Interfaces

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das folgende Kapitel motiviert das Thema dieser Studienarbeit und erläutert den strukturellen Aufbau der Arbeit.

1.1 Motivation

In der Vergangenheit wurde Wissen in erster Linie durch Informationen auf Papier weitergegeben, verarbeitet und archiviert. Für die Verwaltung von Archiven war ein beträchtlicher Aufwand an Personal und Räumlichkeiten notwendig. Die Suche nach Informationen und der Zugriff darauf war relativ aufwendig und nur vor Ort möglich. Die Übermittlung von Informationen auf Papier über die traditionellen Wege der Post und Versanddienste war und ist auch heute noch ein langsamer und kostenintensiver Prozess.

Die rasante Entwicklung der Informations- und Kommunikationstechniken während der letzten Jahrzehnte ermöglicht es heute, riesige Archive von Büchern, Zeitschriften und anderen Dokumenten durch einen ungleich geringeren Aufwand zu pflegen und zu verwalten. Die Speicherung von Informationen auf Papier wird dabei durch digitale Formate ersetzt. Datenbank- und Information-Retrieval-Techniken ermöglichen eine gezielte Suche nach Dokumenten und einen effizienten Zugriff. Archive dieser Art bezeichnet man im Allgemeinen als “digitale Bibliotheken”. Dabei können nicht nur textuelle Dokumente sondern auch multimediale Inhalte wie Bild-, Audio- und Videomaterial archiviert werden. Durch die Entwicklung des Internets spielen geographische Entfernungen beim Zugriff auf all diese Informationen kaum mehr eine Rolle.

Eine neue Qualität des ortsunabhängigen Informationszugriffes entstand in den letzten Jahren durch die Entwicklung des “Mobile Computing” unter Verwendung drahtloser Kommunikationstechniken wie z.B. WLAN (Wireless Local Area Network). Hiermit wird der Zugang zu Dokumenten über Notebooks und Handheld-Geräte von nahezu jedem Ort zu jeder Zeit möglich.

Für den Zugriff auf digitale Bibliotheken sind Applikationen erforderlich, die Anwendern über eine intuitiv zu bedienende Oberfläche ein an das jeweilige Ein-

satzgebiet angepasstes Werkzeug bieten, mit dem nach Dokumenten gesucht und auf diese zugegriffen werden kann. Die Entwicklung einer solchen Software, welche einen mobilen Zugriff auf die Informationen und Dokumente einer bestimmten Anwendungsdomäne ermöglicht, sowie der Aufbau eines Dokumentenservers, welcher die entsprechenden Inhalte zur Verfügung stellt, ist Inhalt dieser Studienarbeit. Als konkretes Anwendungsszenario wird ein Konferenzbesuch verwendet.

1.2 Aufbau der Arbeit

Die Arbeit ist folgendermaßen aufgebaut:

Kapitel 2 - Theoretische Grundlagen erläutert die Begriffe “digitale Bibliothek” sowie “Mobile Computing” und beschreibt das Anwendungsszenario eines Konferenzbesuches.

Kapitel 3 - Anforderungen an die Systemarchitektur beschreibt Anforderungen, welche ein konferenzbegleitendes System hinsichtlich der Systemarchitektur erfüllen muss. Dabei werden Probleme wie Deployment, Laufzeitumgebung, Datenhaltung und Datenzugriff diskutiert.

Kapitel 4 - Systemarchitektur ist das zentrale Kapitel dieser Arbeit. Hier wird eine Architektur für ein System zur Unterstützung von Konferenzbesuchern vorgestellt, die den Anforderungen aus Kapitel 3 genügt. Dabei werden verschiedene Lösungsmöglichkeiten zur Realisierung der einzelnen Systemkomponenten verglichen und die letztendlich eingesetzten Techniken beschrieben.

Kapitel 5 - Visualisierung beschäftigt sich mit dem graphischen Aufbau der Benutzeroberfläche des Nutzer-Frontends für den Zugriff auf die Inhalte der Konferenzdomäne.

Kapitel 6 - Personalisierung erläutert die Personalisierungskonzepte des entwickelten Systems, wie das Erstellen eigener Tagungspläne oder den vorrausschauenden Zugriff auf Dokumente. Dabei werden zunächst Anforderungen diskutiert und anschließend die umgesetzten Konzepte beschrieben.

Kapitel 7 - Zusammenfassung und Ausblick fasst diese Arbeit zusammen und beschreibt Erweiterungen, die auf der Basis dieser Arbeit an dem entwickelten System realisiert werden können.

Kapitel 2

Theoretische Grundlagen

Im ersten Teil dieses Kapitels wird eine kurze Einführung in die Thematik von digitalen Bibliotheken und Mobile Computing geben. Anschließend wird die Anwendungsdomäne für die in dieser Studienarbeit entwickelte Software beschrieben und abschließend werden einige Veröffentlichungen vorgestellt, welche die Entwicklung dieser Arbeit maßgeblich beeinflusst haben.

2.1 Digitale Bibliotheken

Das Wissen einer Nation ist in der heutigen Welt, vor allem für rohstoffarme Länder, zu einem entscheidenden Wirtschaftsfaktor geworden. Das Ausbildungsniveau und die Lernbereitschaft der Bevölkerung, sowie die Verkehrs- und Kommunikations-Infrastruktur, avancierten in den letzten Jahren zu den wichtigsten Standortfaktoren der Industrienationen. Der Zugriff auf Wissen hat daher eine zentrale Bedeutung für unsere Gesellschaft. Eine wichtige Rolle bei der Wissensversorgung der Menschheit spielen seit jeher Bibliotheken. Eine (konventionelle) Bibliothek ist laut Brockhaus “eine planmäßig angelegte Büchersammlung (auch Sammlung von Handschriften, neuerdings auch von audiovisuellem Material), ferner das Gebäude, in dem sie aufgestellt ist”.

Heute steht mehr Wissen zur Verfügung als jemals zuvor in der Menschheitsgeschichte. Auf Grund der ständig steigenden Informationsflut musste in den letzten Jahren nach Alternativen zu konventionellen Informationsträgern gesucht werden. Die moderne Informationstechnik bietet diese Alternativen, mit digitalen Speichermedien und digitalen Speicherformaten. Während konventionelle Bibliotheken in erster Linie Publikationen auf Papier verwalten, besteht der Inhalt digitaler Bibliotheken aus elektronischen Dokumenten. Nach [EF00] ist eine digitale Bibliothek eine Einrichtung:

- die Texte, Bilder, Animationen, Ton- und Videoaufnahmen auf elektronischen Datenträgern vorhält (elektronische Bibliothek),

- die eine Vielzahl von Bibliotheksdiensten in einem ortsübergreifenden Verbund anbietet (virtuelle Bibliothek) und
- deren Bestände und deren Dienste integriert sind, die einen effizienten Zugriff darauf über eine einheitliche Systemoberfläche gestattet und deren Systemintelligenz über die der Teile hinausgeht.

In der Definition werden die häufig synonym verwendeten Begriffe "elektronische Bibliothek", "virtuelle Bibliothek" und "digitale Bibliothek" differenziert. Die Digitalisierung von Medien ergibt die "elektronische Bibliothek" und die Virtualisierung von Bibliotheksdiensten wird als "virtuelle Bibliothek" bezeichnet. Die Integration verschiedener Dienste, die einen effizienten Zugriff auf den Bibliotheksbestand erlauben, führt zur "digitalen Bibliothek". Neben dieser Definition sind in der Literatur weitere zu finden. Eine relativ allgemeine und weithin akzeptierte Definition, die alle wesentlichen Aspekte der meisten anderen Definitionen umfasst, liefert die Digital Library Federation (DLF):

> "Digital libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities." [Wat98]

Die Bedeutung digitaler Bibliotheken liegt vor allem im Potential diverse Probleme, unter denen konventionelle Bibliotheken heute leiden, zu lösen. Dazu zählen unter anderem:

- *Fehlender Stauraum für Bücher:* Digitale Dokumente benötigen weit weniger Platz als gedruckte Dokumente.
- *Mangelnder Platz für Leser:* Während die Größe eines Lesesaals begrenzt ist, können auf den Inhalt digitaler Bibliotheken sehr viele Nutzer gleichzeitig von verschiedenen Orten zugreifen.
- *Säurefraß an Büchern:* Zwar unterliegen digitale Datenträger, wie CDs/DVDs und Magnetbänder, genau wie Papier einem Alterungsprozess, sie können jedoch bei Bedarf einfach vervielfältigt werden.
- *Entfernung zwischen Wohnort eines Nutzers und der Bibliothek:* Die Entfernung zwischen Nutzer und digitaler Bibliothek spielt dank der modernen Kommunikationstechnik kaum noch eine Rolle. Der Zugriff auf digitale Bibliotheken ist über das Internet von nahezu jedem Ort möglich.
- *Vergriffene Bücher:* Da in digitalen Bibliotheken Kopien von Dokumenten verliehen werden, ist jedes Buch der Bibliothek immer verfügbar.

Beispiele für digitale Bibliotheken die z.T. auch öffentlich zugänglich sind, sind unter anderem die ACM Digital Library[1] (Association for Computing Machinery), in der alle Artikel welche in den letzten Jahren bei der ACM veröffentlicht wurden gespeichert sind, sowie der Link-Service des Springer Verlages[2], der wiederum alle Veröffentlichungen dieses Verlages bereitstellt.

2.2 Mobile Computing

Die Fortschritte im Bereich der drahtlosen Kommunikationssysteme und die dazu parallel verlaufende Entwicklung von immer kleineren und leistungsfähigeren Rechnern, ermöglicht heute eine neue Qualität des ortsunabhängigen Informationszugriffs. Über mobile Endgeräte wie Notebooks und PDAs (Personal Digital Assistant) kann von nahezu jedem Ort und zu jeder Zeit mit Hilfe von drahtlosen Netzen auf stationäre Netzwerke, wie auch auf andere mobile Geräte zugegriffen werden. Die Verfügbarkeit mobiler Netzwerke ist dabei in den letzten Jahren deutlich gestiegen, gleichzeitig sind die Preise für notwendige Hardwarekomponenten und Endgeräte gesunken. Eine genaue Definition für den Begriff “Mobile Computing” liefert beispielsweise James B. Zimmerman, University of Maryland:

> ”The term ’Mobile computing’ is used to describe the use of computing devices - which usually interact in some fashion with a central information system - while away from the normal, fixed workplace. Mobile computing technology enables the mobile worker to: (a) create; (b) access; (c) process; (d) store; and (e) communicate information without being constrained to a sinlge location.”

Dabei werden verschiedene Arten von Mobilität unterschieden:

- *Endgerätemobilität:* Ein Benutzer ist genau einem konkreten Gerät zugeordnet, mit dem er sich bewegen und auf Daten und Dienste zugreifen kann. (Bsp. Nutzer mit Handy)

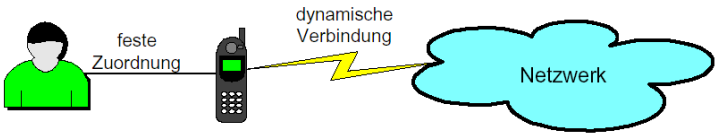

Abbildung 2.1: Endgerätemobilität (Quelle [Rot02b])

- *Benutzermobilität:* Ein Benutzer verwendet verschiedene mobile Geräte zum arbeiten. (Bsp. ein Nutzer mit Handy und Notebook)

[1] http://www.acm.org/dl
[2] http://www.link.springer.de

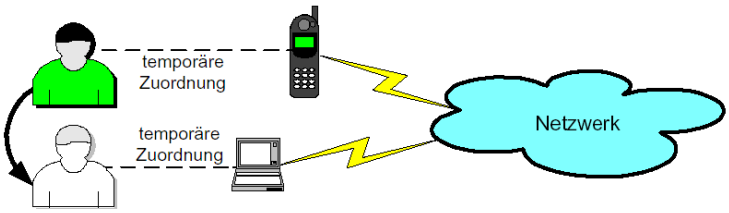

Abbildung 2.2: Benutzermobilität (nach [Rot02b])

Innerhalb dieser Arbeit steht der Mobilitätsbegriff für das erste Szenario, dass heißt das sich ein Anwender zusammen mit seinem mobilen Gerät bewegt.

Abbildung 2.3: Mobile Endgeräte (Quelle [Rot02b])

Die Vielfalt und Leistungsfähigkeit mobiler Endgeräte hat sich in den letzten Jahren genauso rasch Entwickelt, wie die Technologie drahtloser Kommunikation. Eine Auswahl mobiler Geräte ist in Abbildung 2.3 dargestellt. Einige Einschränkungen sind aber bei der Nutzung von mobilen Geräten und vor allem bei der Nutzung von ultraportablen Geräten zu beachten:

- *Akkulaufzeit:* Die Entwicklung von Batterien und Akkus hält kaum mit dem Bedarf der neu entwickelten Geräte Schritt. Grund hierfür sind vor allem große Farbdisplays, Audio und Videoanwendungen, rechenintensive Anwendungen im Allgemeinen und nicht zuletzt auch drahtlose Netzwerkkarten.

- *Speicherplatz:* Da viele mobile Geräte keine Festplatten besitzen, ist die Speicherung großer Datenmengen z.T. schwierig.

- *Zugriff auf große Datenmengen:* Trotzt der Fortschritte der drahtlosen Datenübertragung, haben Netzwerke dieser Art noch nicht die Übertragungsraten konventioneller Netzwerke erreicht. Der Zugriff auf große Datenmengen ist daher mit größeren Zugriffszeiten verbunden.

- *Displaygröße:* Auch wenn Geräte wie PDAs inzwischen über immer größere Displays verfügen, so sind die Bildschirmauflösungen vieler mobiler Geräte gegenüber Desktoprechnern sehr klein. Dies bedeutet für Anwendungsentwickler, dass Applikationen speziell an diese Verhältnisse angepasst werden müssen.

Zu den etablierten Anwendungen, die in den letzten Jahren im Bereich Mobile Computing entwickelt wurden, zählen unter anderem das Messaging in Mobilfunknetzen (SMS, und seit kurzem MMS), Autonavigationssysteme und auch elektronische Touristen- und Museumsführer.

Weitere Ausführungen zum Thema Mobile Computing sowie eine ausführlichere Erläuterung der angerissenen Aspekte, befinden sich z.B. in [Rot02b] und dem dazu erschienenen Buch [Rot02a].

2.3 Beschreibung des Anwendungsszenarios

Das im Rahmen dieser Studienarbeit entwickelte System wurde speziell für den Einsatz innerhalb eines Konferenzbetriebes ausgerichtet. Ziel bei der Entwicklung des Systems war es, Konferenzbesuchern einen mobilen Zugriff auf den Inhalt einer Konferenzdomäne zu ermöglichen.

Eine Konferenz ist gegliedert in Sessions, welche aus einem oder mehreren Vorträgen, Diskussionsrunden oder Meetings bestehen. Die Sessions sind üblicherweise thematisch geordnet und Sessions gleicher oder ähnlicher Thematik werden zu Pfaden organisiert, welche zeitlich parallel verlaufen. Eine Session wird im Allgemeinen von einer Person, dem Session-Chair, moderiert und eine Präsentation oder ein Vortrag wird von mindestens einem Vortragenden präsentiert. Jede dieser Personen hat die Möglichkeit Begleitmaterial zu seiner Präsentation oder der von ihm geleiteten Session bereit zu stellen. Zu den Begleitdokumenten können Abstracts, Foliensätze oder ganze Artikel gehören.

Ein Fallbeispiel verdeutlicht die wichtigsten Funktionen der Client-Applikation, welche in dieser Studienarbeit entwickelt werden soll:

Ein Besucher nimmt an einer Konferenz teil und möchte über sein Notebook die Vorteile digitaler Medien nutzen. Nach der Registrierung am ersten Konferenztag, erhält er die Möglichkeit eine Client-Applikation über das lokale Netzwerk des Konferenzveranstalters auf seinem Notebook zu installieren. Die Applikation bietet Zugriff auf alle Artikel, Foliensätze und sonstige Informationen rund um die Konferenz.

Nachdem der Besucher den Veranstaltungsplan der Konferenz studiert hat, wählt er die für ihn interessanten Vorträge aus. Durch diese Selektion erstellt er sich einen persönlichen Konferenzplan. Anschließend begibt er sich in die erste Präsentation und hat dort die Möglichkeit sich den Foliensatz des Vortragenden über das WLAN des Gebäudes auf sein Notebook zu laden. Außerdem interessiert er sich für weitere Dokumente zum Thema der Vortrags. Über eine Suchmaske gibt der Konferenzbesucher die entsprechenden Stichworte ein und erhält eine Liste von Dokumenten, die er im Anschluss an die Präsentation studieren kann. Falls sich ein Foliensatz oder ein Dokument im Verlauf der Konferenz ändert, wird bei jedem Zugriff automatisch das aktuellste Dokument geladen.

Nachdem der Konferenzbesucher abends wieder im Hotel angekommen ist, wird er an einen am nächsten Tag stattfindenden Vortrag erinnert. Dieser Vortrag

behandelt ein sehr interessantes Thema mit dem sich der Konferenzbesucher schon im Vorfeld eingehender beschäftigen möchte. Dazu öffnet er den vom Vortragenden bereitgestellten Artikel, der den Vortrag ausführlich beschreibt. Da das Hotel nur einen sehr teuren Internetzugang anbietet und das lokale Netzwerk des Konferenzveranstalters nicht verfügbar ist, kann keine Verbindung zum Dokumenten-Server hergestellt werden. Die Client-Applikation greift deshalb auf eine CD des Konferenzveranstalters zurück, welche alle Dokumente der Konferenz-Proceedings enthält. Da der Vortrag vom Konferenzteilnehmer im Veranstaltungsplan als interessant markiert wurde, befinden sich die Vortragsfolien bereits im Cache des Notebooks, so das auch diese offline verfügbar sind.

2.4 Verwandte Arbeiten

Im Folgenden werden einige Arbeiten vorgestellt, die diese Studienarbeit beeinflusst haben.

The Conference Assistant [DSAF99] *The Conference Assistant* ist eine prototypische mobile kontextsensitive Applikation zur Unterstützung von Konferenzbesuchern. Das Team um Anind K. Dey stellte das System auf der dritten ISWC[3] vor. Die Applikation hilft Konferenzbesuchern bei der Auswahl von Präsentationen, versorgt sie mit Informationen während einer Präsentation und ermöglicht das Anfertigen von Notizen. Der Fokus dieser Arbeit liegt auf der Erfassung und Auswertung von Kontextinformationen, die es der Applikation erlauben auf die aktuelle Situation des Nutzers und auf dessen Umgebung zu reagieren. Betritt ein Konferenzbesucher beispielsweise einen Raum, wird automatisch der Name der Präsentation und der Name des Vortragenden angezeigt. Auch der Kontext anderer Personen wird berücksichtigt, so ist es z.B. möglich die Position von Kollegen innerhalb des Konferenzgebäudes zu bestimmen. Dazu wird eine von den Autoren entwickelte Architektur verwendet, die in einem *The Context Toolkit* genannten Framework umgesetzt wurde.

Anders als *The Conference Assistant* ist die im Rahmen dieser Studienarbeit entwickelte Applikation in erster Linie als konventioneller Client für digitale Bibliotheken konzipiert. Eine Erweiterung in Richtung der Erfassung und Auswertung von situationsbeschreibenden Daten soll dabei aber nicht ausgeschlossen werden. Einige Ideen bezüglich der Funktionalität des *Conference Assistant* werden in dieser Studienarbeit aufgegriffen.

Situationsgesteuerter mobiler Zugriff auf digitale Bibliotheken [Haa01] In dieser Diplomarbeit von Peter Haase wird eine Systemarchitektur für den situationsgesteuerten mobilen Zugriff auf digitale Bibliotheken vorgestellt. Als Anwendungsdomäne wird auch hier neben einem Ausstellungsszenario ein Konferenzbesuch betrachtet. Ähnlich wie bei Dey, werden Kontextinformationen erfasst und für

[3] 3rd **I**nternational **S**ymposium on **W**earable **C**omputers, San Fransisco 1999

den situationsgesteuerten Datenzugriff verwendet. Anders als im *Conference Assistant*-System, wo eine ständige Netzwerkverbindung der Clients vorrausgesetzt wird, ist der Mobilitätsbegriff in dieser Arbeit dahingehen erweitert worden, dass ein Anwender auch dann Zugriff zu mindestens auf Teile des Datenbestandes hat, wenn keine Verbindung zum Server besteht. Dies wird durch eine verteilte Speicherung der Daten erreicht. Dieser Ansatz spielt auch in dieser Studienarbeit eine entscheidende Rolle.

Personalisierungskonzepte für digitale Bibliotheken [Zei01] In der ebenfalls an der Universität Rostock erstellten Diplomarbeit von Andre Zeitz, werden verschiede Konzepte zur Personalisierung von digitalen Bibliotheken vorgestellt und in Form von Diensten in die Architektur eines bestehenden Bibliotheksystem eingefügt. Dazu werden zunächst Anforderungen diskutiert, welche derartige Dienste erfüllen müssen. Anschließend wird die Funktionalität der einzelnen Dienste ausgearbeitet und abschließend werden die Dienste zu einer Personalisierungskomponente zusammen gefasst. Einige der vorgestellten Konzepte, wie das Erstellen von Nutzerprofilen, eigene Sichten auf den Datenbestand und die Annotation von Dokumenten wurden in dieser Studienarbeit übernommen.

Kapitel 3

Anforderungen an die Systemarchitektur

Ziel dieses Kapitels ist es Anforderungen zu analysieren, die ein System erfüllen muss, welches Konferenzbesuchern den Zugriff auf die digitalen Inhalte einer Konferenzdomäne in einer mobilen Umgebung ermöglicht.

3.1 Allgemeine Anforderungen

Mobilität Während eines Konferenzbesuches bewegen sich die Teilnehmer ständig zwischen verschiedenen Sessions oder Demonstrationen innerhalb des Veranstaltungskomplexes. Um diese Bewegungsfreiheit nicht einzuschränken, ist ein Datenzugriff mit Hilfe drahtloser Kommunikationsmedien entscheidend. Der Zugriff auf Dokumente und konferenzrelevante Informationen, wie zum Beispiel dem Veranstaltungsplan, darf nicht nur innerhalb bestimmter Räume oder Bereiche möglich sein, sondern muss von jedem Ort des Konferenzgebäudes erfolgen können. Nur so ist es möglich, dass sich ein Besucher während einer Pause in der Cafeteria über den Starttermin der nächsten Session informieren und anschließend in der Präsentation den Foliensatz des Vortragenden auf sein Notebook laden kann.

Dennoch sollte nicht davon ausgegangen werden, dass eine drahtlose Netzwerkverbindung überall besteht. In den wenigsten Fällen wird das lokale Netzwerk des Konferenzveranstalters auch im Hotel eines Konferenzbesuchers zugänglich sein. Es ist aber wünschenswert, dass die Mobilität des Anwenders nicht eingeschränkt wird und bestimmte Daten auch in diesem Fall zur Verfügung stehen. Die Art der Datenspeicherung muss insofern auf diesen Wunsch reagieren, als dass ein Zugriff zu mindestens auf Teile des Datenbestandes auch offline möglich ist.

Plattformunabhängigkeit Da nicht vorrausgesetzt werden kann, dass alle Besucher einer Konferenzveranstaltung über die selbe Hardware- und Softwareausstattung verfügen, ist eine plattformunabhängige Realisierung des System-Frontends wichtig. Eine Client-Applikation sollte eine möglichst breite Palette verschiedener

Betriebssysteme auf verschiedenen Endgeräten unterstützen. In einer mobilen Umgebung bedeutet dies vor allem die Unterstützung von Notebooks und der gängigsten PDA's.

Deployment Die Installation eines Nutzer-Frontends sollte für den Anwender möglichst unkompliziert sein. Der Installationsprozess sollte durch einen grafischen Assistenten begleitet werden, der die Auswahl der verschiedenen Installationsparameter erlaubt. Zum Abschluss der Installation sollte die Anwendung in den Desktop des Nutzer integriert werden, so dass das Starten der Applikation nicht über "kryptische" Aufrufe auf der Kommandozeile erfolgen muss.

Des weiteren sollten die Softwarevoraussetzungen für die Installation einer Client-Applikation möglichst gering sein. Die Akzeptanz einer Anwendung wird sicherlich nicht dadurch gesteigert, dass für deren Betrieb die Installation eines Relationalen Datenbank Systems, oder ähnlich komplexer Software, notwendig ist.

3.2 Datenhaltung

Die Art und Weise der Datenspeicherung ist entscheidend für die Effizienz des Datenzugriffs und für die Verfügbarkeit des Datenbestandes. Im Folgenden soll erläutert werden, welchen Anforderungen die Speicherung der Daten gerecht werden muss, um ein möglichst gutes Systemverhalten zu erreichen.

Das Datenaufkommen in einem digitalen Bibliotheken System, wie es in dieser Studienarbeit entwickelt wird, kann nach [Haa01] folgendermaßen klassifiziert werden:

- Unstrukturierte und semistrukturierte Multimedia-Daten, wie Volltexte, Bilder, Videos, Audiodaten etc.;
- Daten, welche die multimedialen Inhalte der Bibliothek beschreiben;
- Daten, die das konkrete Anwendungsszenario beschreiben (z.B. einen Veranstaltungsplan);
- Daten, welche die momentane Situation des Bibliotheksbenutzers beschreiben.

Unstrukturierte und semistrukturierte Daten Bei unstrukturierten und semistrukturierten Daten einer Konferenzdomäne handelt es sich vor allem um wissenschaftliche Artikel, Foliensätze oder Abstracts. Diese Dokumente bilden den eigentlichen Inhalt der digitalen Bibliothek und werden zunächst zentral gespeichert. Je nach Umfang des Datenbestandes ist mehr oder weniger Aufwand zur Verwaltung dieser Daten notwendig, um beim Zugriff möglichst kurze Antwortzeiten zu gewährleisten. Im Allgemeinen sollten hier Content Management Systeme zum Einsatz kommen.

Damit der Zugriff, zu mindestens auf einige der Dokumente auch dann möglich ist wenn keine Netzwerkverbindung zum Server besteht, ergibt sich die Notwendigkeit, Dokumente auch auf den lokalen Laufwerken der Clients verfügbar zu machen. Je nach Umfang der Daten wäre es beispielsweise denkbar, dass der Konferenzveranstalter einen Teil der Dokumente auf CDs oder DVDs an die Konferenzbesucher verteilt. Der Inhalt dieser Medien könnte dann als Datenbasis verwendet werden, die unabhängig von der Erreichbarkeit des Serversystems zur Verfügung steht.

Eine andere Möglichkeit zum Offline-Zugriff auf Dokumente bieten Caching-Mechanismen, dabei werden angeforderte Dokumente zwischengespeichert. Öffnet der Benutzer ein Dokument, so wird dieses transparent auf einem lokalen Datenträger gespeichert, so dass es zukünftig auch ohne Netzwerkverbindung geladen werden kann. Um eine möglichst hohe Trefferquote für den Dokumentencache zu erreichen, besteht außerdem die Möglichkeit, nicht nur vom Benutzer angeforderte Dokumente zu cachen, sondern mit Hilfe von Prefetching-Strategien bereits Dokumente in den Zwischenspeicher zu laden von denen erwartet wird, dass der Anwender diese in naher Zukunft anfordern wird. Im Kapitel über Personalisierung wird später erläutert werden, welche Informationen zur Umsetzung solcher Prefetching-Techniken genutzt werden können.

Beide Verfahren, sowohl das Caching als auch das Erstellen einer Datenbasis auf CD/DVD, verbessern nicht nur die Offline-Funktionalität der Client-Applikation, sondern tragen auch zur Verringerung des Netzwerkverkehrs bei, da die Dokumente nicht jedes mal neu bzw. überhaupt nicht vom Server übertragen werden müssen.

Metadaten der multimedialen Inhalte Die zweite Klasse von Daten beinhaltet Dokument beschreibende Informationen wie Titel, Autor oder Erscheinungsjahr. Mit Hilfe dieser Daten können Suchanfragen auf den Inhalt der digitalen Bibliothek gestellt werden. Um eine effiziente Suche zu ermöglichen müssen diese Daten in Indexform gespeichert werden. Dies sollte im Allgemeinen implizit durch das zur Dokumentenspeicherung verwendete Content Management System erfolgen, kann aber auch durch eigene Strukturen realisiert werden. Soll eine Metadatensuche auch ohne eine Serververbindung möglich sein, müssen auch diese Daten auf die Clients repliziert werden. Von den Dokumenten des Suchergebnisses kann dann wenigstens auf die Dokumente zugegriffen werden, die sich im lokalen Cache des Client-Systems befinden.

Daten des Anwendungsszenario Neben dem eigentlichen Inhalt der digitalen Bibliothek und den Metadaten der verwalteten Dokumente entstehen in verschiedenen Anwendungsszenarien weitere Daten. Im Falle eines Konferenzbetriebes sind dies vor allem Raumpläne des Konferenzgebäudes, Zeitpläne der einzelnen Sessions sowie aktuelle Informationen über die Veranstaltung. Diese Daten sollten auf jeden Fall auf den Client-Systemen gespeichert werden, da sie für den Besucher in

nahezu jeder Situation relevant sind und damit ständig verfügbar sein müssen.

Situationsbeschreibende Daten Eine letzte Klasse von Daten ergibt sich aus Informationen, welche die momentane Situation oder das Umfeld des Anwenders beschreiben. Hierzu gehören z.B. die aktuelle Position des Konferenzbesuchers innerhalb des Konferenzgebäudes oder die Position von Kollegen aber auch der Name der aktuellen Präsentation oder des Sprechers. Wichtig bei der Erfassung derartiger Daten ist eine hinreichende Genauigkeit, vor allem bei Positionsbestimmungen. Dies erfordert in der Regel einen entsprechenden Aufwand an Hardware.

Aus der Analyse der einzelnen Datenklassen folgt, dass eine zentrale Datenspeicherung in einer mobilen Umgebung nicht ausreichend ist. Je nach Datentyp müssen einzelne Daten lokal auf den Client-Systemen gecached oder repliziert werden. Die Gründe hierfür sollen abschließend noch einmal zusammengefasst werden:

- *Offlinebetrieb:* In Situationen in denen der Nutzer über keine Netzwerkverbindung verfügt, sollen zu mindestens Teile der Daten dennoch verfügbar sein.
- *Effizienz:* Durch eine verteilte Datenhaltung, in der Daten auf den lokalen Laufwerken der Client-Systeme gespeichert werden, kann der Datenzugriff durch eine geringere Zahl von Netzwerkzugriffen beschleunigt werden.

In [Haa01] wird sogar noch ein dritter Grund für eine Datenverteilung angeführt:

- *Datenschutz:* Bei persönlichen Daten ist es unter Umständen gar nicht gewünscht diese serverseitig zu speichern.

3.3 Datenzugriff

In einem typischen Client/Server-System einer digitalen Bibliothek erfolgt eine zentrale Speicherung der Daten auf einem Server. Die Clients greifen ausschließlich auf eine Datenquelle zu, dem Zentralserver. Im vorherigen Kapitel wurden Gründe erläutert, warum in dem mobilen Szenario eines Konferenzbetriebes, eine alleinige zentrale Datenspeicherung nicht ausreichend ist. Die daher notwendige Verteilung der Daten muss beim Zugriff auf Dokumente und andere Inhalte berücksichtigt werden.

Datenzugriffe in einer verteilten Umgebung folgen nicht mehr nur dem oben beschriebenen Muster. Ein Dokument kann sich bei einer verteilten Speicherung auf verschiedenen Datenträgern, in verschiedenen Versionen auf verschiedenen Rechnern befinden. Beim Zugriff auf Dokumente muss vor allem festgestellt werden, von welcher Quelle die Daten bezogen werden können. Ziel dabei ist es, in

Abhängigkeit vom Systemzustand die möglichst aktuellste Datenquelle zu ermitteln. Entscheidendes Kriterium ist dabei die Verfügbarkeit des Netzwerkes. Ist keine Netzwerkverbindung vorhanden, muss auf die lokalen Datenträger des Client-Systems zurückgegriffen werden, dabei wird in Kauf genommen das eventuell veraltete Daten geladen werden. Eine verfügbare Netzwerkverbindung bedeutet dagegen nicht unbedingt, dass ein angefordertes Dokument immer vom Server geholt werden muss. Liegt ein Dokument beispielsweise in der gleichen Version sowohl im lokalen Cache als auch auf dem Server vor, so muss dies erkannt und zur Effizienzsteigerung die lokale Version verwendet werden.

3.4 Anfragen

Grundsätzlich kann in digitalen Bibliotheken zwischen zwei klassischen Anfragearten unterschieden werden. Diese müssen durch ein konferenzbegleitendes System, wie es in dieser Studienarbeit entwickelt wird, unterstützt werden:

- *Suche auf Metadaten:* Dazu gehört die Suche nach Dokumenten mit Hilfe von Metadaten wie Autor oder Titel.
- *Information Retrieval:* Dies bezeichnet die Suche auf unstrukturierten Daten. Im Konferenzszenario handelt es sich dabei in erster Linie um Volltexte. Es sind allerdings auch Anfragen auf Multimedia-Dokumente, welche Audio- oder Videodaten enthalten, denkbar.

Werden durch das System auch situationsbeschreibende Daten erfasst, dann ist außerdem ein dritter Anfragetyp möglich:

- *Inferenzanfragen:* Dies sind Anfragen, welche von den Client-Systemen in Abhängigkeit von der aktuellen Situation und Umgebung des Nutzers automatisch gestellt werden. Ist dem System beispielsweise die Position eines Konferenzbesuchers bekannt, so kann in Abhängigkeit von der Uhrzeit ermittelt werden in welcher Session er sich gerade befindet.

Kapitel 4

Systemarchitektur

Im Folgenden Kapitel wird eine Systemarchitektur vorgestellt, welche nach den oben beschriebenen Anforderungen entworfen wurde. Dabei werden verschiedene Problemlösungen verglichen und die letztendlich verwendeten Techniken beschrieben.

Die Architektur ist in einem System umgesetzt worden, welches zur Begleitung der VLDB 2003[1] in Berlin eingesetzt wurde. Das System besteht aus einer *eClient* getauften Java-Applikation und zwei Backendservern. Die Client-Applikation ermöglicht es Konferenzbesuchern mobil auf die Dokumente der Konferenz-Proceedings zuzugreifen. Des Weiteren werden konferenzrelevante Informationen, wie der Veranstaltungsplan, elektronisch verfügbar gemacht. Der Datenzugriff ist auch bei einer unterbrochenen Netzwerkverbindung, zumindest auf Teile des Dokumentenbestandes, möglich.

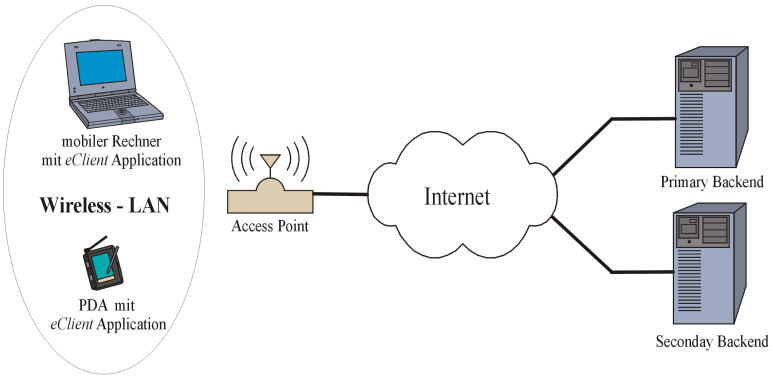

Abbildung 4.1: Systemarchitektur Überblick

[1] **V**ery **L**arge **D**ata **B**ases, internationale Konferenz rund um das Thema Datenbanktechnologie - www.vldb.org

4.1 Überblick

Abbildung 4.1 gibt einen groben Überblick über die Architektur des *eClient*-Systems. Die Client-Geräte sind über ein drahtloses lokales Netzwerk mit dem Internet verbunden und können so auf die Inhalte der Konferenzdomäne zugreifen. Um die Verfügbarkeit zu erhöhen werden die Daten redundant auf zwei Backendsystemen gespeichert. Die Backends befanden sich im Falle der VLDB in Rechenzentren verschiedener Einrichtungen. Da es unwahrscheinlich war, dass beide Systeme gleichzeitig ausfallen, wurde auf diese Art eine gewisse Ausfallsicherheit geschaffen.

4.2 Drahtlose Kommunikation

Für die drahtlose Anbindung der mobilen Endgeräte an das Internet sind verschiedenen Techniken einsetzbar. Grundsätzlich können drahtlose Netzwerktechnologien in drei Klassen eingeteilt werden:

- *Zellulare Netze:* Hierbei handelt es sich um flächendeckende zellulare Netze, die bundesweit verfügbar sind und Roaming auch über Landesgrenzen hinweg unterstützen. Zellulare Netze gehören zu den Wide Area Networks (WAN). Zu den etablierten Techniken zählen GSM[2] und die darauf aufbauende Dienste HSCSD[3], GPRS[4] und EDGE[5]. Die Übertragungsraten liegen dabei vergleichsweise niedrig bei 115,2 KBit/s für HSCSD, 171,2 KBit/s für GPRS und 473,6 KBit/s für EDGE, wobei dies bereits theoretische Maximalwerte sind, die sich durch Kanalbündelung ergeben. In der Praxis werden Werte dieser Art nicht erreicht. Die tatsächlich nutzbaren Bandbreiten liegen weit niedriger. Nokias kürzlich vorgestellte “mobile Surfstation” 7700 empfängt Daten über HSCSD beispielsweise mit maximal 43,2 KBit/s und über GPRS mit maximal 53,6 KBit/s.

  Hieraus ergeben sich einige Einschränkungen für die Nutzung dieser Dienste bzgl. der Übertragung von großen Multimedia-Daten. Dieses Manko könnte in naher Zukunft durch den Aufbau von UMTS[6]-Netzen zu mindestens gelindert werden, da bei UMTS Übertragungsraten von bis zu 2 MBit/s möglich sein werden. Dies gilt allerdings nur im Nahbereich von sogenannten “Piko Zellen” mit einer Größe zwischen 50 und 100 Meter. Auf mittleren und langen Strecken sinken die Übertragungsraten auch hier auf 384 KBit/s (Mikro Zelle) bzw. 144 KBit/s (Makro Zelle).

- *Wireless LANs (Local Area Networks):* Wireless LANs sind mit einer Reichweite von bis zu 300 Meter gut für den Einsatz innerhalb von Gebäuden oder

[2] **G**lobal **S**ystem for **M**obile Communications
[3] **H**igh **S**peed **C**ircuit **S**witched **D**ata
[4] **G**eneral **P**acket **R**adio **S**ervice
[5] **E**nhanced **D**ata Rates for **G**SM **E**volution
[6] **U**niversal **M**obile **T**elecommunications **S**ystem

auf Firmengeländen geeignet. Als Standards sind hier vor allem IEEE 802.11 und HiperLAN zu nennen, wobei letzterer trotz einiger einzigartiger Merkmale bis jetzt keine nennenswerte praktische Bedeutung erreichen konnte. Die Bandbreiten des IEEE 802.11 Standards liegen je nach Ausprägung zwischen 1 und 54 MBit/s, womit diese Technik auch zur Übertragung größerer Datenmengen gut geeignet ist. Die verfügbare Bandbreite eines Zugangsknotens (Access Points) wird allerdings auf alle Nutzer des Access Points aufgeteilt. Greift eine große Anzahl Nutzer gleichzeitig über den selben Access Point auf das Netzwerk zu, kann es auch hier zu Engpässen kommen. Am weitesten verbreitet sind im Moment Geräte, welche die 802.11b Spezifikation erfüllen und Übertragungsraten von bis zu 11 MBit/s ermöglichen.

- *Wireless PANs (Personal Area Networks) und BANs (Body Area Networks):* Netzwerke dieser Kategorie erlauben eine kabellose Datenübertragung über kurze und ultra kurze Strecken. Die erfolgreichsten Vertreter dieser Netzwerkkategorie sind Bluetooth und IrDA[7]. Bluetooth basiert auf Funktechnologie, IrDA dagegen nutzt Infrarotlicht zur Datenübertragung. Dabei ist zu beachten, dass zur Datenübertragung mittels IrDA “Sichtkontakt” zwischen den am Datenaustausch beteiligten Geräten bestehen muss und nur jeweils zwei Geräte gleichzeitig (Punkt zu Punkt Verbindung) Daten austauschen können. Die Datenübertragungsraten von IrDA liegen je nach Standard zwischen 115,2 KBit/s (SIR[8]) und immerhin 16 MBit/s (VFIR[9]). Die Reichweite ist allerdings auf maximal ein Meter beschränkt. Bluetooth erlaubt Datenübertragungsraten von bis zu 723,2 KBit/s bei einer Reichweite von bis zu zehn Meter bei 1 mW Sendeleistung und bis zu 100 Meter bei 100 mW. Dabei können maximal acht Geräte zu einem sogenannten Pikonetz zusammengeschlossen werden. Bis zu zehn Pikonetze können wiederum ein Scatternetz bilden, wenn sich die jeweils benachbarten Pikonetzte überschneiden.

Eine ausführlichere Beschreibung aller Technologien befindet sich beispielsweise in [Rot02b]. Entscheidend bei der Auswahl einer Übertragungstechnologie sind vor allem die durchschnittliche Reichweite und Datenübertragungsrate eines Verfahrens, aber auch die Verbreitung und die Verfügbarkeit von entsprechenden Geräten. Bei der Verwendung von Wide Area Netzwerken müssen zusätzlich auch die Kosten, welche dem Anwender durch die Nutzung von Diensten externer Provider entstehen, berücksichtigt werden.

Auf der VLDB 2003 wurde vom Konferenzveranstalter, für die Verbindung der *eClient*-Geräte zum Internet, ein WLAN nach dem IEEE802.11b Standard zur Verfügung gestellt. Ein entscheidender Vorteil dieser Technologie ist deren weite Verbreitung. In vielen mobilen Geräten neuerer Generation sind entsprechen-

[7] **I**fra**r**ed **D**ata **A**ssociation
[8] **S**erial **I**nfra**r**ed
[9] **V**ery **F**ast **I**nfra**r**ed

de Adapter bereits integriert. Da die Kommunikation zwischen Client-Applikation und Server lediglich auf dem HTTP-Protokoll aufsetzt, wäre aber auch der Einsatz anderer Netzwerktechnologien möglich gewesen.

Interessante Möglichkeiten bietet die Kopplung verschiedener Netzwerktechnologien. Denkbar wäre z.B. die Nutzung eines WLAN's nach IEEE 802.11 und eines WAN mit den Eigenschaften wie sie in Zukunft UMTS bieten wird. Dies ermöglicht einen mobilen Datenzugriff innerhalb des Konferenzgebäudes über das lokale Netzwerk des Veranstalters und außerhalb des Gebäudes über das WAN eines externen Providers, jeweils mit akzeptablen Datenübertragungsraten.

4.3 Geräteplattformen

Ein Ziel bei der Entwicklung der *eClient*-Applikation war die Unterstützung einer möglichst großen Vielfalt von Geräte- und Systemplattformen. Für die Entwicklung von plattformunabhängigen Anwendungen existieren zur Zeit zwei konkurrierende Techniken. Zum einen ist dies die Java 2 Plattform der Firma Sun Microsystems, zum anderen das .NET Framework von Microsoft. In beiden Programmierumgebungen wird aus dem Programmquellcode nicht direkt nativer Maschinencode, sondern zunächst ein Zwischencode erzeugt. Dieser Zwischencode ist plattformunabhängig und kann auf jedem System ausgeführt werden, für das eine entsprechende Laufzeitumgebung existiert. Laufzeitumgebungen für das .NET Framework von Microsoft existieren bisher jedoch lediglich für Windows Systeme, sowie für FreeBSD. Da dies auf Grund von ungeklärten Lizenzfragen voraussichtlich bis auf Weiteres so bleiben wird, wurde für die Entwicklung der *eClient*-Applikation der Programmiersprache Java der Vorzug gegeben. Hierfür existieren bereits seit langem Laufzeitumgebungen für eine große Anzahl von Systemen, unter anderem Windows, Macintosh, Linux sowie für verschiedene Unixvarianten einschließlich Solaris, AIX, HP-UX und Tru64. Durch verschiedene Varianten der Java-Plattform werden Geräte vom Mobiltelefon bis hin zum Enterprise-Server unterstützt. Eine Übersicht über die verschiedenen Java-Editionen gibt Abbildung 4.2.

Für das *eClient*-System kamen als Zielgeräte vor allem Notebooks, Tablet PCs und PDAs in Frage. Auf Grund des engen Zeitrahmens, der für die Entwicklung des Systems zur Verfügung stand, wurde entschieden die *eClient*-Applikation ausschließlich für Geräte zu entwickeln, die durch die Java 2 Standard Edition unterstützt werden. Getestet wurde die Applikation auf verschiedenen Notebooks unter Windows 2000/XP, Linux und Solaris. Notebooks bieten durch ihre typischen Geräteeigenschaften wie Speicher- und Displaygröße sowie durch ihre weite Verbreitung gute Vorraussetzungen zum Einsatz im beschriebenen Anwendungsszenario.

Geräte wie PDAs, welche durch die Java 2 Micro Edition abgedeckt werden, wurden aus Zeitgründen zunächst nicht berücksichtigt. Bei einer Portierung der *eClient*-Applikation müssten die speziellen Eigenschaften dieser Geräte wie z.B.

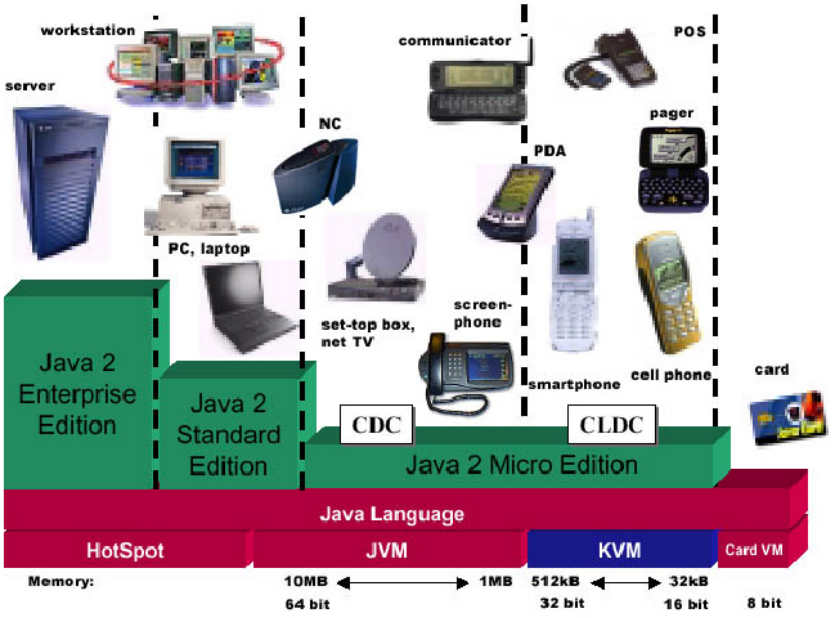

Abbildung 4.2: Die Java-Varianten und ihre Zielgeräte (Quelle [Haa01]

der geringere Speicherausbau und die kleineren Displays berücksichtigt werden. Außerdem müssten einige Bibliotheken, welche die jetzige *eClient*-Applikation benötigt, ersetzt werden, da diese unter der Java 2 Micro Edition nicht zur Verfügung stehen. Dies betrifft vor allem den Volltextindizierer *Lucene*, sowie den zum Deployment verwendeten JNLP-Client *Java Web Start*.

4.4 Datenhaltung

Im Folgenden Kapitel wird erläutert, wie die Inhalte der einzelnen Datenklassen, welche im Abschnitt 3.2 eingeführt wurden, im *eClient*-System verwaltet und gespeichert werden.

Unstrukturierte und semistrukturierte Daten Der eigentliche Inhalt der digitalen Bibliothek besteht im *eClient*-System aus den Artikeln und den Foliensätzen der Präsentationen, sowie der Webseite des Konferenzveranstalters. Bei der verteilten Speicherung der Daten, wird davon ausgegangen, dass Änderungen an den Daten ausschließlich auf den Servern erfolgen. Der Zugriff der Client-Systeme beschränkt sich also auf das Auslesen der Bibliotheksinhalte. Die Daten werden auf die folgenden drei Datenquellen verteilt:

- *CD:* Als Datenbasis wird vom Konferenzveranstalter eine CD produziert, welche die Artikel der Präsentationen sowie die Homepage des Konferenz-

veranstalters, jeweils in der zum Produktionszeitpunkt aktuellen Version enthält. Diese CD dient als primäre Datenquelle, sowohl für den Offline- als auch den Online-Zugriff.

- *Server:* Als Backendsysteme werden zwei HTML-Server verwendet, auf welchen sich der komplette Datenbestand in der aktuellsten Version befindet. Im Gegensatz zur CD, können die Dokumente auf den Servern vor, während und nach der Konferenz aktualisiert werden.

- *Client:* Eine Teilmenge der Dokumente auf den Servern, wird in das lokale Dateisystem der Clients gecached. Dies ermöglicht einen Offline-Zugriff auf aktualisierte Dokumente der Server, auf die bereits einmal zugegriffen wurde, welche sich auf Grund ihrer Aktualität jedoch nicht auf der CD befinden.

Metadaten der multimedialen Inhalte Die Verwaltung der Metadaten der Dokumente, wie auch der Dokumente selbst, kann durch ein Content Management System erfolgen. Da der Umfang der Metadaten und Dokumente im Falle der VLDB 2003 aber überschaubar war, wurde darauf verzichtet und für die Bereitstellung der Dokumente wurden, wie oben beschrieben, HTML-Server eingesetzt. Daraus ergab sich die Notwendigkeit, eigene Indexe für eine Metadaten- und Volltextsuche aufzubauen. Für die Bewältigung dieser Aufgabe wurde das freie System *Lucene* verwendet.

Lucene ist Teil des Apache Jakarta Projekts und komplett in Java entwickelt. Das System stellt Klassen zur Erzeugung und Abfrage von Volltextindexen zur Verfügung. Als Indexform werden invertierte Listen eingesetzt. Die Indexe werden als Dateien gespeichert und unterstützen verschiedene linguistische Operationen wie Stopworteliminierung und Stemming. Bei der Indizierung werden für jedes Dokument Felder angelegt, welche den Dokumententext oder verschiedene Metadaten wie Autor, Titel oder Erscheinungsjahr enthalten. Anschließend kann über die Felder nach Dokumenten gesucht werden. Im *eClient*-System werden zwei Indexstrukturen erzeugt:

- *Paper-Index:* Der Paper-Index enthält die Informationen der pdf-Dokumente der Konferenz-Proceedings. Neben den Volltextinformationen werden zusätzlich die URL des Dokuments, der Dokumententitel, die Namen der Autoren, der Name der Session, innerhalb welcher das Dokument präsentiert wird, sowie die Kategorie der Session als Metadaten gespeichert.

- *HTML-Index:* In einem zweiten Index wird die Homepage des Konferenzveranstalters indexiert, so dass auch der Webauftritt der Konferenz über die *eClient*-Applikation durchsucht werden kann.

Ein Vorteil, den der Aufbau eigener Indexstrukturen bietet, ist die einfache Replikation dieser Daten. Beide Indexe werden sowohl auf den HTML-Servern abgelegt, als auch bei der Installation der Client-Applikation lokal auf den mobilen Geräten

gespeichert. Damit sind die Indexstrukturen grundsätzlich offline verfügbar und können auch ohne Internetverbindung genutzt werden. Bei Änderungen am Dokumentenbestand werden die Indexstrukturen auf den Servern aktualisiert. Beim nächsten Online-Zugriff werden die Änderungen dann auf die Client-Systeme übertragen.

Daten des Anwendungsszenario Bei den Daten des Anwendungsszenarios handelt es sich vor dem Hintergrund eines Konferenzbetriebes vor allem um den Veranstaltungsplan. Dieser sollte immer auf den Client-Systemen verfügbar sein, da er ständig benötigt wird. Die Speicherung dieser Daten soll im Folgenden genauer erläutert werden.

Da der Veranstaltungsplan einer Konferenz im Allgemeinen nicht mehr als 5 - 10 Tage umfasst, wurde darauf verzichtet diese Daten in einem Datenbanksystem zu speichern. Die Daten werden stattdessen direkt in einem XML-Dokument verwaltet, welches immer lokal auf den Client-Systemen verfügbar ist. Beim Starten der *eClient*-Applikation wird das XML-Dokument mit Hilfe eines SAX-Parsers analysiert und so der Veranstaltungsplan extrahiert. Die Performance des Parse-Prozesses war bei dem Veranstaltungsplan der VLDB 2003, bei dem etwa 60 Sessions mit 170 Präsentationen sowie 400 Personen verwaltet werden mussten, akzeptabel.

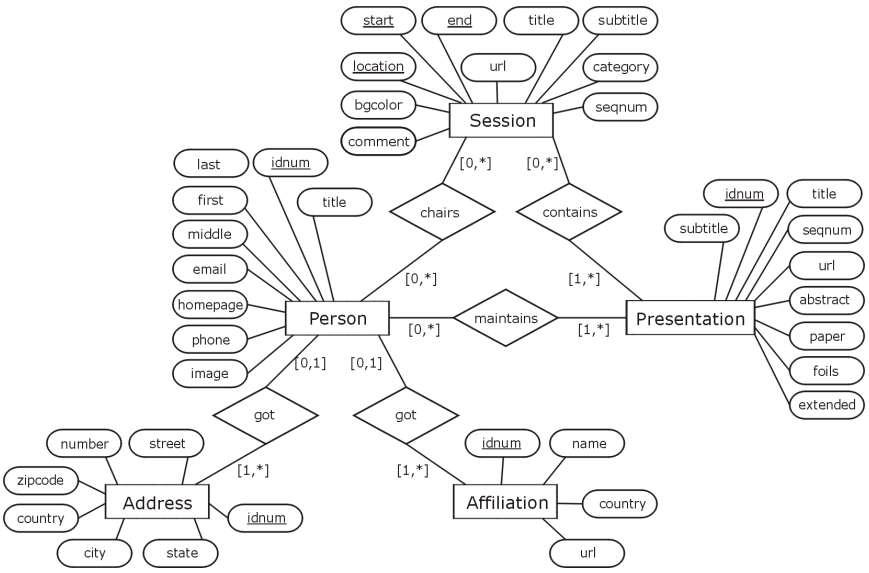

Abbildung 4.3: Entity Relationship Modell eines Konferenzveranstaltungsplanes

Für die Entwicklung der DTD, welche die Struktur des XML-Dokumentes beschreibt, dass zur Speicherung des Veranstaltungsplans im *eClient*-System verwendet wird, diente ein abstraktes Datenmodell als Vorlage. Die Entwicklung einer DTD aus einem gegebenen Entity Relationship Modell (ER-Modell) ist unter an-

derem in [Dol03] beschrieben.

Abbildung 4.3 zeigt ein ER-Modell für den Veranstaltungsplan einer Konferenz. Ein Veranstaltungsplan besteht demnach aus Sessions. Eine Session kann durch eine oder mehrere Personen, den Session-Chairs, begleitet werden. Einer Session können Präsentationen zugeordnet sein, welche innerhalb der Session vorgetragen werden. Jeder Präsentation ist auf jeden Fall eine Session zugeordnet und jede Präsentation wird durch mindestens eine Person präsentiert. Zu einer Person kann eine Adresse sowie eine Einrichtung angegeben sein, bei der die Person beschäftigt ist.

Das ER-Modell enthält ausschließlich 1:n und n:m Beziehungen. Die n:m Beziehungen hätten durch unabhängige Beziehungselemente mit jeweils zwei Referenzen auf die an der Beziehung beteiligten Elemente umgesetzt werden müssen. Da diese Beziehungen jedoch keine eigenen Attribute haben, wurden sie wie auch die 1:n Beziehungen auf einfache ID/IDREF-Beziehungen abgebildet. Dies verbessert die Lesbarkeit des entstehenden XML-Dokuments. Die aus dem ER-Modell entwickelte Konferenz-DTD ist in Abbildung 4.4 dargestellt. Die Darstellung wurde allerdings auf die Elemente sowie deren ID/IDREF-Attribute reduziert.

```
<!ELEMENT Conference (Address*, Affiliation*, Person*,
Presentation*, Session*)>
<!ELEMENT Session (EMPTY)>
  <!ATTLIST Session presentations IDREFS #IMPLIED>
  <!ATTLIST Session chair IDREFS #IMPLIED>
<!ELEMENT Presentation EMPTY>
  <!ATTLIST Presentation idnum ID #REQUIRED>
  <!ATTLIST Presentation performers IDREFS #REQUIRED>
<!ELEMENT Person EMPTY>
  <!ATTLIST Person idnum ID #REQUIRED>
  <!ATTLIST Person address IDREF #IMPLIED>
  <!ATTLIST Person affiliation IDREF #IMPLIED>
<!ELEMENT Affiliation EMPTY>
  <!ATTLIST Affiliation idnum ID #REQUIRED>
<!ELEMENT Address EMPTY>
  <!ATTLIST Address idnum ID #REQUIRED>
```

Abbildung 4.4: DTD zur Speicherung eines Konferenzveranstaltungsplanes

In Anwendungsszenarien die einen größeren Datenumfang mit sich bringen, wäre es aber durchaus sinnvoll den Veranstaltungsplan in einem Datenbanksystem auf einem Server zu verwalten, da die Datenverwaltung in einem XML-Dokument dann uneffizient wird. Ein Anwender könnte in diesem Fall bei einer bestehenden Netzwerkverbindung den gesamten Veranstaltungsplan vom Datenbankserver abfragen und die für ihn interessanten Veranstaltungen auswählen. Aus diesen Infor-

mationen kann dann wiederum ein relativ kleines XML-Dokument generiert werden, dass auf das Client-System übertragen wird und dann für die Offline-Arbeit zur Verfügung steht.

Situationsbeschreibende Daten Auf die Erfassung von situationsbeschreibenden Daten wurde im Rahmen des *eClient*-Systems für die VLDB 2003 verzichtet. Hinsichtlich der zukünftigen Weiterentwicklung des Systems wäre die Erfassung und Auswertung derartiger Daten aber sicherlich eine sinnvolle Erweiterung.

Eine der wichtigsten Informationen die diesbezüglich in einem Konferenzsystem erfasst werden kann, ist die Position der Konferenzbesucher. Mit Hilfe dieser Information lassen sich einige interessante Funktionen umsetzten, wie sie z.B. in [DSAF99] beschrieben sind. So ist es unter anderem möglich, einem Konferenzteilnehmer automatisch den Titel der aktuellen Präsentation anzuzeigen, wenn dieser einen Raum betritt. Bei Verfahren für die Bestimmung von Positionsdaten wird zwischen zwei Systemkategorien unterschieden:

- *Tracking:* Hierbei wird die Position eines Objektes von einem Sensorennetzwerk erfasst und zentral auf einem Rechner gespeichert. Alle Objekte müssten dafür spezielle Marken (engl. Tag, Badge) tragen, die von den Sensoren erkannt werden können. Die Position eines Objektes kann dann auf Anfrage vom Server übermittelt werden.
- *Positioning:* Andererseits wäre es denkbar, dass die Client-Systeme die Positionsdaten selber ermitteln. Hierzu fängt das bewegte Objekt Signale von Sendern oder Baken auf. Der Vorteil dieses Verfahren ist, dass die Positionsdaten direkt beim Client anfallen.

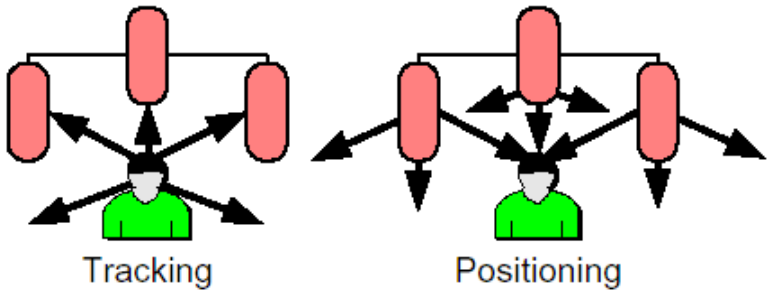

Abbildung 4.5: Verfahren zur Positionsbestimmung (Quelle [Rot02b])

Für die konkrete Realisierung eines Verfahrens zur Positionsbestimmung können verschiedene Techniken eingesetzt werden. Drei davon sollen hier kurz beschrieben werden:

- *Cell of Origin (COO):* Die Reichweite von Funksignalen ist begrenzt. Dieser Umstand kann zur Bestimmung der Position eines beweglichen Objektes genutzt werden. Ist ein Objekt in der Lage das Signal eines bestimmten Senders

zu empfangen, kann daraus geschlussfolgert werden, dass sich das Objekt in der Nähe dieses Senders befindet.

- *Time of Arrival (TOA), Time Difference of Arrival (TDOA):* Da Signale eine bestimmbare Ausbreitungsgeschwindigkeit haben (Funk z.B. 300.000 km/s), kann durch die Messung von Signallaufzeiten verschiedener Sender die Position eines Objektes relativ präzise bestimmt werden.

- *Messung der Signalstärke:* Die Stärke eines Signals nimmt mit zunehmender Entfernung von der Signalquelle ab. Die Signalstärke ist also ein Maß für die Entfernung vom Sender. Auch diese Eigenschaft kann zur Positionsbestimmung genutzt werden.

Der Aufbau einer Infrastruktur für die Bestimmung von Positionsdaten bedeutet in der Regel einen erheblichen Aufwand, daher ist es sinnvoll, nach Möglichkeit die bestehende Infrastruktur auszunutzen. Ein bereits vorhandenes WLAN das beispielsweise zur Anbindung mobiler Geräte an des Internet dient, kann gleichzeitig auch zur Positionsbestimmung verwendet werden. Diese Technik wurde z.B. im Nibble-Projekt ([nib]) prototypisch umgesetzt. In einer Trainingsphase wird an jedem möglichen Ort eines Gebäudes oder Gebäudekomplexes, die Signalstärke der Basisstationen des WLANs gemessen. Bei der Verwendung des Systems wird dann einem Satz gemessener Signalstärken, der Ort zugeordnet, der am besten zu dem gemessenen Datensatz passt. Bei Tests konnte mit dem System eine Genauigkeit von 2-3 m erreicht werden. Ein Nachteil des Systems ist allerdings die aufwendige Trainingsphase. Diese muss bei Veränderungen der Umgebung sogar wiederholt werden, da die Signalstärke durch Gegenstände beeinflusst werden kann.

Eine ausführliche Beschreibung der vorgestellten Techniken, sowie weitere konkrete Verfahren zur Positionsbestimmung befinden sich in [Rot02b].

4.5 Datenzugriff

Zugriff auf Dokumente Wird ein Dokument als Ergebnis einer Anfrage ermittelt, muss entschieden werden, von welcher Datenquelle der verteilten Architektur das Dokument bezogen wird. Die Datenquellen im *eClient*-System wurden im letzten Kapitel beschrieben und sind in Abbildung 4.6 dargestellt.

Die “best Source” Ermittlung erfolgt in Abhängigkeit des Online-Status des mobilen Endgerätes, sowie des Datums der letzten Änderung des angeforderten Dokuments auf den verschiedenen Datenquellen. Wird ein Dokument durch ein Client-System angefordert, wird zunächst überprüft, ob eine Verbindung zu einem der HTML-Server aufgebaut werden kann. Ist dies nicht der Fall, kann nur mit den lokalen Daten im Cache oder auf der CD gearbeitet werden. Kann eine Verbindung zu einem der Server aufgebaut werden, so muss überprüft werden, auf welcher Datenquelle sich die aktuellste Version des angeforderten Dokumentes befindet. Dies erfolgt durch einen Vergleich der Zeitstempel der Dokumente. Für den Fall, dass

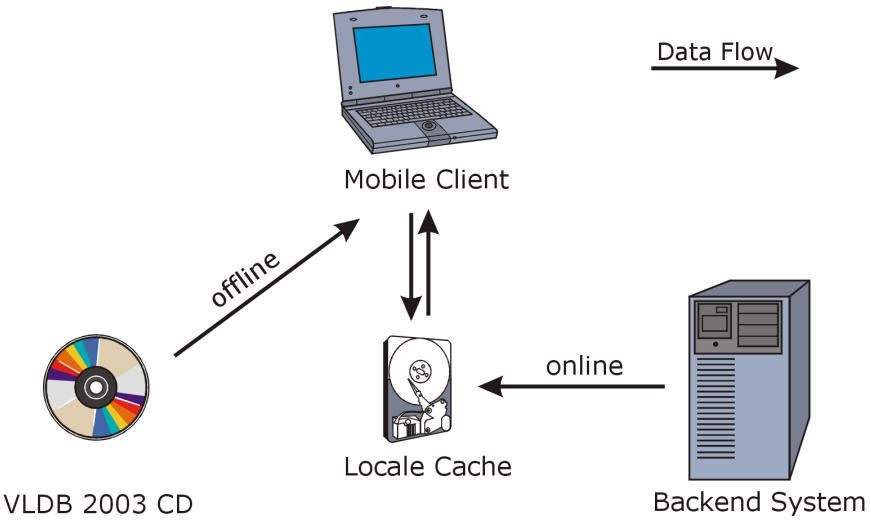

Abbildung 4.6: Datenquellen und Datenfluss im *eClient*-System

die Dokumentenversion des Servers aktueller als die Version des lokalen Speichers ist, wird das Dokument vom Server in den Cache übertragen. Von dort kann es dem Benutzer anschließend angezeigt werden. Abbildung 4.7 zeigt den Ablaufplan eines Dokumentenzugriffs im *eClient*-System nach dem beschriebenen Verfahren.

Da der Cache eines Client-Systems auf Grund von Kapazitätsbeschränkungen im Allgemeinen nicht alle Dokumente des Servers aufnehmen kann, muss eine Ersetzungsstrategie verwendet werden. Dies bedeutet, dass Dokumente im Cache nach einer bestimmten Regel ersetzt werden, falls neue Dokumente eingelagert werden sollen, der Cache aber gewisse Grenzen überschreitet. Kriterien die zu einer Dokumentverdrängung führen können, sind beispielsweise:

- die Überschreitung der maximalen Größe des Caches,
- die Überschreitung der maximalen Anzahl von Objekten im Cache.

Um das Dokument zu ermitteln, welches als nächstes aus dem Cache verdrängt werden soll, existieren verschiedenste Verfahren. Ziel aller Verfahren ist es, mit Hilfe von Heuristiken zu ermitteln, welches der Dokumente im Zwischenspeicher am Wahrscheinlichsten nicht mehr benötigt wird und damit ersetzt werden kann. Dazu werden die Dokumente zunächst nach einem Schlüssel sortiert, über die Sortierung bestimmt die Ersetzungsstrategie anschließend das Dokument, welches als nächstes aus dem Zwischenspeicher verdrängt wird. Als Sortierkriterium lassen sich folgende Eigenschaften eines Dokuments nutzen:

- der Zeitpunkt der Einlagerung in den Cache,
- der Zeitpunkt der letzten Referenzierung,

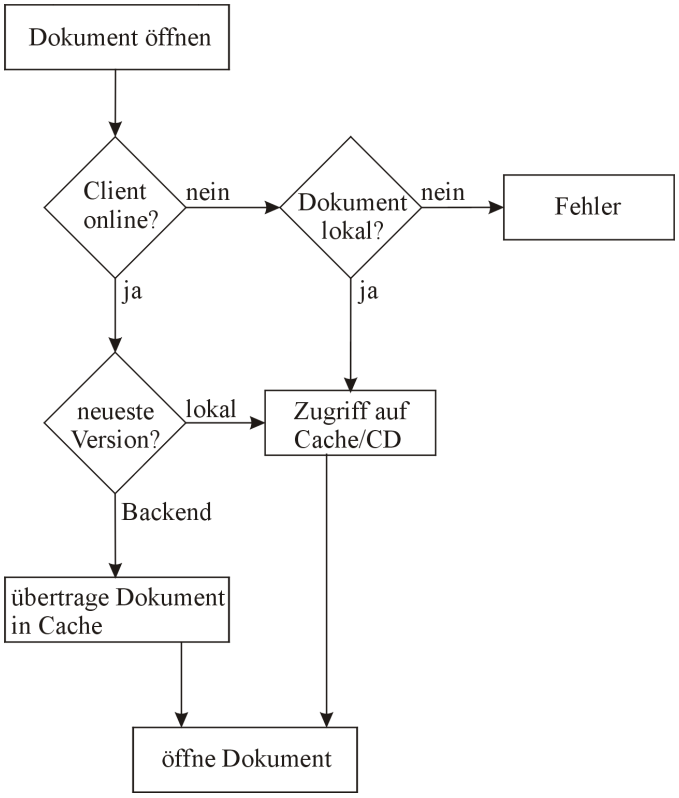

Abbildung 4.7: Zugriff auf Dokumente im *eClient*-System

- die Anzahl der Referenzen,
- die Größe eines Dokuments.

Durch die Sortierung, nach einem oder mehreren dieser Schlüssel, ergibt sich die Ersetzungstrategie. Einen Überblick über verschiedene Ersetztunstrategien liefert [Jon99]. Im Folgenden sind drei gängige Verfahren kurz beschrieben:

- *FIFO:* Die *First in First out* -Strategie ersetzt das Dokument, welches als erstes in den Cache eingelagert wurde. Interessant ist dieses Verfahren vor allem durch seine einfache Implementierung.
- *LFU:* Der *Least Frequently Used* -Mechanismus entfernt das Dokument mit der geringsten Anzahl von Zugriffen.
- *LRU:* Die *Least Resently Used* -Strategie verdrängt das Dokument, welches am längsten nicht mehr referenziert wurde.

Zum Zeitpunkt der VLDB 2003 wurde aus Zeitgründen noch keine Ersetzungsstrategie in der *eClient*-Appliaktion implementiert. Dies sollte in zukünftigen Versionen aber behoben werden. Die Auswahl einer “optimalen” Strategie ist allerdings

schwierig. Für das konkrete Anwendungsszenario eines Konferenzbesuches sind vor allem die LFU- und LRU-Strategie interessant, da so häufig bzw. kürzlich genutzte Dokumente ständig im Cache bleiben und damit für den Konferenzbesucher immer offline verfügbar sind. Um die Effizienz der Verfahren zu steigern, sollten zusätzlich Informationen, die das Anwendungsszenario liefert, berücksichtigt werden. Hierfür können z.B. der Tagungsplan einer Konferenz sowie das Nutzerprofil eines Anwenders genutzt werden. Gehört ein Dokument zu einer Präsentation, die in der nächsten Zeit stattfindet oder für die sich ein Besucher besonders interessiert, dann sollte dieses Dokument bevorzugt behandelt werden.

Zugriff auf Tagungsprogramm und Index Da sowohl das Tagungsprogramm als auch die Indexstrukturen zur Dokumentensuche bei der Installation der Client-Applikation auf den Client-Systemen gespeichert werden, sind diese Daten grundsätzlich offline verfügbar. Der Zugriff erfolgt daher immer über das lokale Dateisystem. Ändert sich das Tagungsprogramm oder eine der Indexstrukturen auf dem Server, so werden diese Änderungen automatisch auf die Client-Systeme übertragen, dabei werden alte Daten durch neue überschrieben. Die *eClient*-Applikation implementiert diesen Mechanismus nicht selbst, sondern nutzt zur Aktualisierung dieser Daten den JNLP-Client *Java Web Start*.

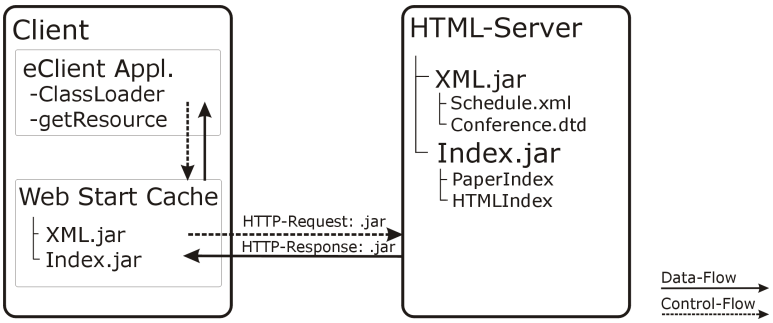

Abbildung 4.8: Zugriff auf das Tagungsprogramm und die Indexe

JNLP ist ein Protokoll zum Deployment von Java-Applikationen, welches im Folgenden Kapitel ausführlich beschrieben wird. Um die Daten für *Java Web Start* verfügbar zu machen, werden die Indexstrukturen und das Tagungsprogramm in jar-Archiven verpackt. Bei jedem Start der Client-Applikation wird durch *Java Web Start* überprüft, ob die Daten auf dem Server geändert wurden. Ist dies der Fall wird das entsprechende jar-Archiv neu auf das Client-System übertragen. Über die Klasse ClassLoader ist es möglich, mit Java direkt auf Dateien in einem jar-Archiv zuzugreifen, so dass die Archive nicht auf den Client-Systemen entpackt werden müssen. In Abbildung 4.8 ist der Zugriff auf das Tagungsprogramm und die Indexstrukturen mit Hilfe von *Java Web Start* dargestellt.

4.6 Deployment und Wartung

Prototypische Implementationen, wie sie gewöhnlich im Rahmen von Studien- und Diplomarbeiten entwickelt werden, dienen lediglich Demonstrationszwecken. Bei Systemen, die für den praktischen Einsatz bestimmt sind, wie dies beim *eClient*-System der Fall ist, muss zusätzlich das Problem der Softwareauslieferung an den "Kunden", in diesem Fall der Konferenzbesucher, gelöst werden. Der Installationsprozess eines Nutzer-Frontends sollte möglichst unkompliziert sein und durch ein grafisches Installationstool unterstützt werden. Der Vorgang der Auslieferung und Installation einer Software wird auch als Deployment bezeichnet. Zum Deployment von Java-Applikationen sind mehrere Wege möglich, die im Folgenden kurz beschrieben werden.

JAR-Archive Die einfachste Möglichkeit besteht aus dem Erstellen von Java-Archiven. Hierfür werden alle Ressourcen einer Anwendung wie Klassen, Property-Files, Bilder oder Icons in ein jar-Archiv gepackt und an den Anwender ausgeliefert. Dies kann beispielsweise über das Internet erfolgen. Das zum Erstellen des Archivs notwendige Tool ist Bestandteil des Java Software Development Kits (SDK). Falls eine Java Laufzeitumgebung (JRE) installiert ist und im jar-Archiv festgelegt wurde, welche der Klassen die main()-Methode enthält, kann eine so verpackte Anwendung auf der Kommandozeile durch den Aufruf von "java -jar jar-file" gestartet werden. Dies ist allerdings für "normale Nutzer" ein recht komplizierter Weg. Einfacher wird es, wenn der Dateityp jar in der Desktopumgebung entsprechend verknüpft ist. In diesem Fall reicht ein simpler Mausklick auf das Java-Archiv, um die entsprechende Anwendung zu starten. Die Verknüpfung von Dateitypen ist allerdings durch ein Installationsskript schwer zu realisieren, da je nach verwendetem Betriebssystem unterschiedliche Maßnahmen dafür notwendig sind.

Startskripte und Launcher Eine weitere Möglichkeit Java-Applikationen zu starten bieten Startskripte oder Launcher. Letztere können zum Beispiel über das Java Nativ Interface (JNI) erstellt werden. Sun liefert mit dem Software Development Kit mehrerer Header-Dateien aus, welche die Generierung von Launchern auf C/C++ Basis ermöglichen. Damit können z.B. für Windows Nutzer exe-Dateien erzeugt werden, die Java-Applikationen starten. Ähnliche Ergebnisse können mit Shellskripten oder bat-Dateien erreicht werden. Dem Anwender wird es so erspart, das Programm über eine Konsole von Hand zu starten. Das Problem der eigentlichen Installation der Java-Application wird damit aber nicht gelöst.

Installer Eine umfassende Lösung zur Installation von Applikationen und deren Integration in den Desktop bieten Installer. Es existieren diverse kommerzielle Systeme, die Installationsprogramme sehr komfortabel nach gewählten Vorgaben generieren. Die generierten Installer ermöglichen es dem Nutzer, während der Installation beispielsweise nach vorhandenen Java Laufzeitumgebungen zu suchen

oder eine Laufzeitumgebung zu installieren, falls noch keine vorhanden ist sowie Startmenüeinträge oder Icons auf dem Desktop anzulegen. Ein Nachteil dieser Methode ist aber, dass für jede Zielplattform ein eigener Installer generiert werden muss. Außerdem sind die Kosten für derartige Systeme beträchtlich. Freie Systeme wie *InstallAnywhere Now!* von Zero G[10] sind eher selten und meistens in ihrer Funktionalität eingeschränkt.

Java Web Start und JNLP Alle bisher vorgestellten Lösungen haben jeweils einige Nachteile. Eine letzte Möglichkeit zur Installation von Java-Applikationen bietet das Java Network Launching Protocol (JNLP). JNLP ist eine von Sun Microsystems vorgeschlagene Spezifikation[11], mit Protokollen und APIs, zum Deployment von Java-Applikationen über das Internet. *Java Web Start* ist Sun's JNLP-Client Referenz Implementation. Der *Java Web Start* Anwendungsmanager ist Bestandteil aller neueren Versionen des Java Runtime Environments und damit auf den meisten Systemen verfügbar, auf denen Java-Applikationen ausgeführt werden können. Das System ermöglicht die Installation von Java-Applikationen über das Internet und gewährleistet ein automatisches Update, so dass immer die neueste Programmversion ausgeführt wird. Neben Sun's Referenzimplementation existieren weitere JNLP-Clients, wie z.B. *OpenJNLP*[12] und *NetX*[13].

Zum Deployment einer Java-Applikation über JNLP, erstellt das Entwicklerteam der zu installierenden Anwendung eine Webseite, die einen Link auf eine jnlp-Datei enthält. In dieser Datei werden im XML-Format alle Informationen angegeben, die zum Download der Programmbestandteile und zum Starten der Applikation notwendig sind. Ein Ausschnitt aus der jnlp-Datei der *eClient*-Applikation ist in Abbildung 4.9 angegeben. Man sieht den Pfad zur Codebase auf dem Server der Universität Rostock, die einzelnen jar-Archive der Applikation und den Namen der main-Klasse. Außerdem wird festgelegt, dass zum Start der Anwendung Java mindestens in der Version 1.4 vorhanden sein muss. Ist dies nicht der Fall wird die Java Laufzeitumgebung bei der Installation automatisch aktualisiert.

Klickt ein Anwender, beim Besuch einer Webseite, auf einen Hyperlink der auf ein jnlp-Datei zeigt, wird diese an die Anwendung übergeben, welche für den entsprechenden Mime Type "application/x -java -jnlp -file" und/oder die Dateiendung ".jnlp" registriert ist. Im Allgemeinen ist dies javaws[.exe]. *Java Web Start* cached darauf hin die notwendigen Ressourcen der Applikation in ein lokales Verzeichnis und führt die Anwendung von dort aus. Der gesamte Installationsprozess kommt, bis auf den initialen Mausklick auf den jnlp-Link, ohne Nutzerinteraktion aus, da nicht einmal ein Installationsverzeichnis festgelegt werden muss. Abschließend kann die Anwendung auf Wunsch in den Desktop integriert werden, in dem ein Eintrag im Startmenü oder ein Icon auf der Arbeitsfläche angelegt wird. Die

[10] http://www.zerog.com
[11] http://www.jcp.org/jsr/detail/56.jsp
[12] http://openjnlp.sourceforge.net
[13] http://ocd.sourceforge.net/netx/netxPage.html

```
<?xml version='1.0' encoding='UTF-8'?>
<jnlp
  spec='1.0+'
  codebase='http://wwwdb.informatik.uni-rostock.de/vldb2003/
            e-client/jars'
  href='eClient.jnlp'>
  <information>
   <title>VLDB 2003 eClient</title>
   <vendor>University of Rostock, Germany</vendor>
  </information>
  <resources>
   <j2se version='1.4+'
    href='http://java.sun.com/products/autodl/j2se'/>
   <jar href='eClient.jar'/>
   <jar href='XML.jar'/>
   <jar href='index.jar'/>
   <jar href='html.jar'/>
  </resources>
  <application-desc main-class='vldb.gui.eClient'/>
</jnlp>
```

Abbildung 4.9: Ausschnitt aus dem eClient.jnlp File

Anwendung ist mit den Daten aus dem Web Start Cache auch offline jederzeit ausführbar. Bei einer bestehenden Internetverbindung prüft *Java Web Start* zusätzlich bei jedem Start, ob neue Versionen, der in der jnlp-Datei angegebenen Ressourcen, bereitstehen. Ist dies der Fall, werden die entsprechenden Dateien nachgeladen.

Der Installationsprozess ist in Abbildung 4.10 an Hand der *Java Web Start* Beispiel-Applikation Draw noch einmal dargestellt. Der Pfeil auf der rechten Seite verdeutlicht was der Benutzer sieht, er klickt auf den Link der Webseite, anschließend startet die Applikation. Der linke und der untere Pfeil zeigen die Abläufe im Hintergrund. *Java Web Start* wird durch den Webbrowser aktiviert und zeigt einen Splashscreen mit dem Download-Status der Applikationsbestandteile an. Alle notwendigen Dateien werden gecached und die Applikation wird gestartet.

Auf Grund der vielen Vorteile der JNLP-Technologie, die im Folgenden noch einmal zusammengefasst werden sollen, wurde diese Technik zum Deployment der *eClient*-Applikation ausgewählt und hat sich im praktischen Einsatz bewährt.

- JNLP-Clients sind kostenlos.
- Der *Java Web Start*-Client ist Bestandeil des Java Runtime Environments von Sun Microsystems und damit auf den meisten Java-fähigen Rechnern

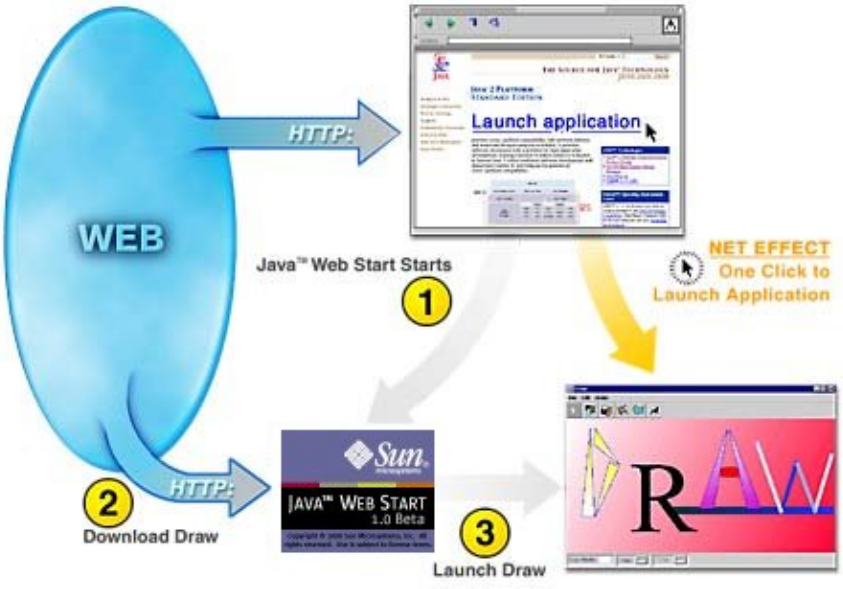

Abbildung 4.10: Installation mit *Java Web Start* (Quelle [WSH]

verfügbar.

- "Single Click Install"
- Anwendungen können auch offline genutzt werden.
- Unabhängig vom Zielsystem der Installation muss der Anwendungsprogrammierer lediglich eine Webseite mit einer jnlp-Datei bereitstellen.
- Die installierte Anwendung wird auf Wunsch in den Desktop integriert.
- Alle angegebenen Ressourcen werden automatisch aktualisiert.
- Sandboxprinzip: Ähnlich wie bei Java Applets, die in einer sicheren Umgebung auf einem Webserver ausgeführt werden, ist dies auch auf Desktoprechnern möglich.

Alle Klassen und sonstigen Ressourcen einer Applikation müssen, um sie für einen JNLP-Client verfügbar zu machen, in jar-Archive verpackt werden. Anschließend werden die Archive elektronisch signiert, um sie vor Manipulationen zu schützen. Die *eClient*-Applikation besteht aus vier jar-Archiven:

- *eClient.jar:* enthält die Java-Klassen der Applikation
- *XML.jar:* beinhaltet das Konferenztagungsprogramm im XML-Format sowie die dazu gehörige DTD

- *index.jar:* enthält die *Lucene*-Indexstrukturen
- *html.jar:* enthält einige html-Seiten

Durch diese Modularisierung wird es *Java Web Start* ermöglicht, einzelne Teile der Applikation zu aktualisieren, so dass nicht jedes Mal alle Dateien neu auf die Client-Systeme übertragen werden müssen. Nach Änderungen an den Indexstrukturen wird beispielsweise nur das index.jar Archiv neu übertragen.

Darüber hinaus bietet das JNLP die Möglichkeit von inkrementellen Updates. Dazu werden die jar-Archive versioniert. Bei Änderungen an den Daten werden die jar-Archive dann nicht komplett neu übertragen, sondern nur die Änderungen. Weitere Informationen zu der JNLP-Technik befinden sich unter anderem in [Jol] und [Bau02] oder auf der *Java Web Start* Seite [WSG] von Sun.

In Abbildung 4.11 ist die Rolle von *Java Web Start* in der *eClient*-Systemarchitektur dargestellt.

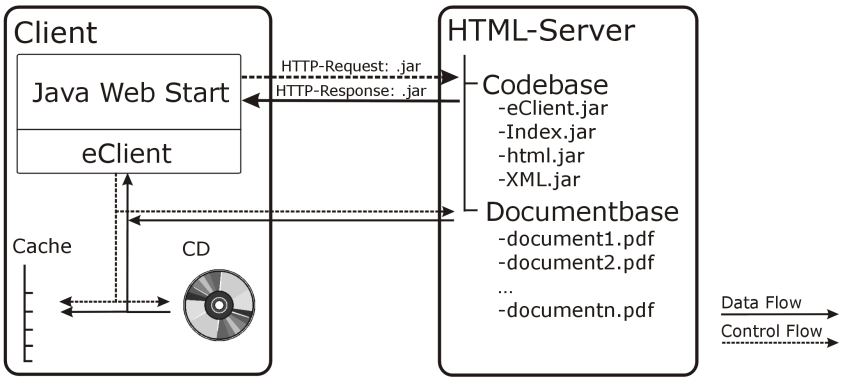

Abbildung 4.11: *eClient*-Systemarchitektur und *Java Web Start*

Kapitel 5

Visualisierung

In diesem Kapitel wird die Repräsentation der Daten der Konferenzdomäne durch das grafische Nutzer-Frontend erläutert.

5.1 Anforderungen

Die *eClient*-Applikation bildet die Schnittstelle zwischen dem Nutzer und den Inhalten der verschiedenen Datenquellen. Für die Entwicklung der grafischen Oberfläche galten zunächst die allgemein gültigen Vorgaben zum Aufbau von Graphical User Interfaces (GUIs), wie sie z.B. in [Eng03] beschrieben sind. Eine Oberfläche sollte klar strukturiert, übersichtlich und intuitiv zu bedienen sein. Sie sollte sich möglichst nahtlos in die vom Benutzer verwendete Umgebung einpassen und angemessen auf Fehleingaben reagieren. Um die Akzeptanz der Anwendung bei den Nutzern zu steigern, muss die Funktionalität des Systems außerdem leicht zu erlernen sein.

Informationsgewinnung Nach [EF00] kann eine Benutzeroberfläche für digitale Bibliotheken drei Arten der Informationsgewinnung unterstützen:

- *Suchen:* Dabei wird zwischen der Suche auf Metadaten und der Suche auf semi- oder unstrukturierten Dokumenten unterschieden. Der Nutzer spezifiziert hierbei eine Anfrage durch einen Anfrageausdruck oder mit Hilfe einer Suchmaske und erhält als Ergebnis eine Liste von Dokumenten.

- *Navigieren:* Mit Navigieren ist der Zugriff auf Daten mit Hilfe der Struktur der Bibliotheksinhalte gemeint. Hierfür werden häufig Baumstrukturen oder Klassifikationsschemata verwendet.

- *Stöbern:* Als Stöbern oder Browsen wird das nicht zielgerichtete Suchen bezeichnet.

Level of Detail In der Regel ist es sinnvoll, nicht in jeder Ansicht alle Informationen, die über ein Objekt verfügbar sind anzuzeigen. Wird beispielsweise eine Liste von Dokumenten dargestellt, ist es ausreichend, wenn nur der Dokumentenname in der Liste erscheint. Interessiert sich ein Nutzer für ein spezielles Dokument, können weitere Informationen bis hin zum kompletten Inhalt angezeigt werden. Dieses Prinzip ist als Level of Detail bekannt. In [Haa01] werden drei Granularitätsstufen vorgeschlagen:

- *Label:* Ein Label ist ein menschenverständlicher Bezeichner für ein Objekt in Form eines Literals, für ein Buch kann dies beispielsweise der Buchtitel sein.
- *Metadaten:* Unter Metadaten werden alle strukturierten Daten, welche ein Objekt näher beschreiben, zusammengefasst. Für ein Buch sind dies z.B. Informationen wie Autor, Verlag und Erscheinungsjahr.
- *Dokument:* Dabei handelt es sich um ein semi- oder unstrukturiertes Dokument, im Falle eines Buches kann dies beispielsweise eine pdf-Datei sein.

5.2 Umsetzung

Zur Entwicklung von Oberflächen für Java-Applikationen existieren zur Zeit drei relevante Bibliotheken:

- *Abstract Windowing Toolkit (AWT):* Das AWT wurde 1996 im Rahmen des JDK1.0 von Sun Microsystems entwickelt. Das Toolkit stellt ein plattformunhabhängiges API zur Entwicklung von Oberflächen zur Verfügung, mit dem portable Applikationen entwickelt werden können. Die konkrete Implementierung der einzelnen GUI-Komponenten ist allerdings plattformabhängig. Jeder GUI-Komponente ist eine Hilfsklasse (peer) zugeordnet, die mit nativem Code direkt das equivalente GUI-Elemente des jeweiligen Betriebssystems anspricht.

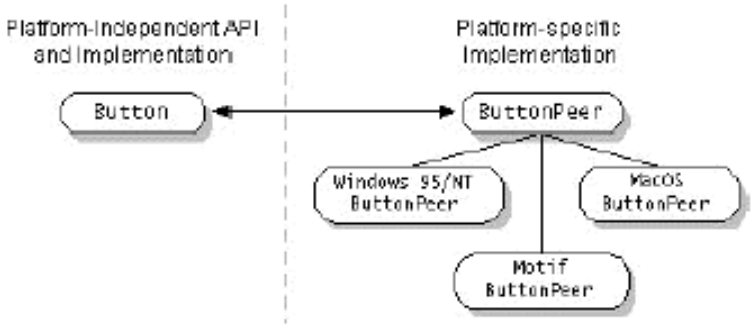

Abbildung 5.1: Funktionsprinzip des AWT (Quelle [AWT]

Eine Oberfläche die mit dem AWT erstellt wurde, hat so die selbe Optik und das selbe Verhalten, wie eine native Anwendung des verwendeten Betriebssystems. Das größte Problem dieser Technik ist, dass nicht alle GUI-Elemente von jeder Plattform unterstützt werden. Das AWT beinhaltet daher lediglich die Elemente, welche auf allen Plattformen vorhanden sind. Elemente wie Tree und Table sind in der Bibliothek nicht enthalten.

- *Swing:* Die *Swing*-Bibliothek wurde 1998 von Sun nachgeliefert, um neben dem relativ beschränkten AWT eine bessere Bibliothek zur Entwicklung von Java-Oberflächen zur Verfügung zu stellen und ist seit dem ebenfalls fester Bestandteil des JDKs. *Swing* setzt auf dem AWT auf, verfolgt aber grundsätzlich einen anderen Weg zur Darstellung der GUI-Elemente. Zwar werden die Top-Level-Fenster weiterhin nach der oben beschriebenen Technik erzeugt, alle anderen GUI-Komponenten werden aber durch Java selbst gezeichnet und nicht durch das jeweilige Betriebssystem dargestellt. Dadurch haben *Swing*-Oberflächen auf jedem Betriebssystem das gleiche Aussehen und das gleiche Verhalten und sind völlig plattformunabhängig. Über verschiedene Look&Feel's kann die Darstellung der Bedienelemente bei Bedarf dem jeweiligen Betriebssystem angepasst werden. Außerdem ist es möglich eigene Komponenten zu erstellen, die in Optik und Verhalten völlig frei definiert werden können. *Swing* hat aber gegenüber dem AWT Nachteile bezüglich der Performance, da die selbstgezeichneten Komponenten erheblich langsamer dargestellt werden, als die Komponenten des Betriebsystems.

- *Standard Widget Toolkit (SWT):* Das SWT ist eine noch sehr junge Bibliothek zum Gestalten von Oberflächen, welche 2001 im Rahmen des Eclipse-Projektes[1] entstand. SWT kann als Kombination der Techniken von AWT und *Swing* betrachtet werden. Wie im AWT wird auch im SWT zur Darstellung der GUI-Elemente mit Hilfe von nativem Code direkt auf Betriebssystem-Ressourcen zugegriffen. Existiert ein GUI-Element auf einem Betriebssystem nicht, so wird dieses Element selbst gezeichnet. Wie in *Swing* können auch im SWT eigene GUI-Elemente erstellt werden. Das SWT versucht so, die Performance-Vorteile des AWT, mit der Flexibilität der *Swing*-Bibliothek zu verbinden. Ein Nachteil von SWT ist aber, dass entsprechende Anwendungen ausschließlich auf Plattformen laufen, für die eine entsprechende SWT-Version implementiert wurde. Außerdem ist das SWT nicht im JDK enthalten, dass heißt, anders als bei *Swing*-Applikationen muss die SWT-Bibliothek vom Anwender vor dem Start der Applikation zusätzlich zur Java Laufzeitumgebung installiert werden, falls dies nicht bereits geschehen ist.

Weitere Informationen und Literaturverweise zu den vorgestellten Bibliotheken befinden sich unter anderem in [Dib01]. Die Oberfläche der *eClient*-Applikation wur-

[1] http://www.eclipse.org

de mit der *Swing*-Bibliothek realisiert. Gründe hierfür waren erstens, dass *Swing* fester Bestandteil des JDKs ist, zweitens die gegenüber dem SWT bessere Kompatibilität zwischen verschiedenen Plattformen und drittens die im Entwicklerteam vorhandene Erfahrung mit der *Swing*-Bibliothek.

Navigierender Zugriff Die *eClient*-Oberfläche ist in zwei Bereiche unterteilt. Das erste Panel hat das Layout eines Organizers und dient dem navigierenden Datenzugriff.

Für die Konferenzdomäne ergibt sich folgende Objekthierarchie: Einer Konferenz sind Tage und einem Tag verschiedene Sessions zugeordnet. Eine Session beinhaltet Metadaten und Präsentationen. Eine Präsentation besitzt ebenfalls Metadaten und zusätzlich Dokumente. Entlang dieser Hierarchie kann navigiert werden.

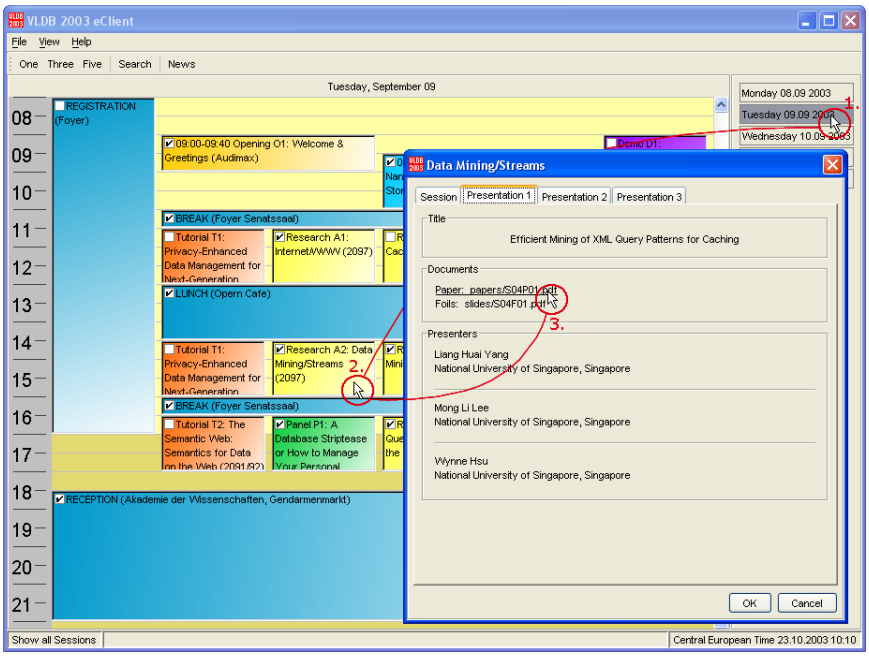

Abbildung 5.2: Navigierender Datenzugriff

Abbildung 5.2 zeigt die Organizer-Ansicht des *eClients*. Die Sessions eines Tages sind in Abhängigkeit von ihrem Beginn und Ende auf der Oberfläche angeordnet. Sessions eines Tracks sind außerdem untereinander dargestellt und Sessions einer bestimmten Kategorie sind farblich einheitlich markiert. Über die Auswahl eines Konferenztages und einer Session kann zu einer gewünschten Präsentation navigiert werden. Ein Dialog gibt Auskunft über die ausgewählte Session und deren Präsentationen. Auf dem Präsentations-Panel befinden sich Verweise zu den Dokumenten.

Suche Der zweite Bereich der Oberfläche dient der Volltextsuche über den Proceeding-Dokumenten und der Homepage des Konferenzveranstalters. Über ein Textfeld kann eine Anfrage abgesetzt werden, welche mit Hilfe der *Lucene*-Indexstrukturen eine Liste relevanter Dokumente zurückliefert. Für die Formulierung der Anfrage wird die *Lucene* eigene Syntax verwendet. Diese unterstützt unter anderem boolesche Verknüpfungen, Phrasen-Suche und unscharfe (Fuzzy-) Suche. Einen kompletten Überblick über die Syntax gibt [Luc03].

Die Liste der Ergebnisdokumente wird sortiert ausgegeben. Je relevanter ein Dokument bezüglich der Anfrage ist, desto besser ist seine Position in der Ergebnisliste. Dieses Verfahren wird als Ranking bezeichnet. Die Funktion, welche hierfür verwendet wird, ist ebenfalls in [Luc03] beschrieben.

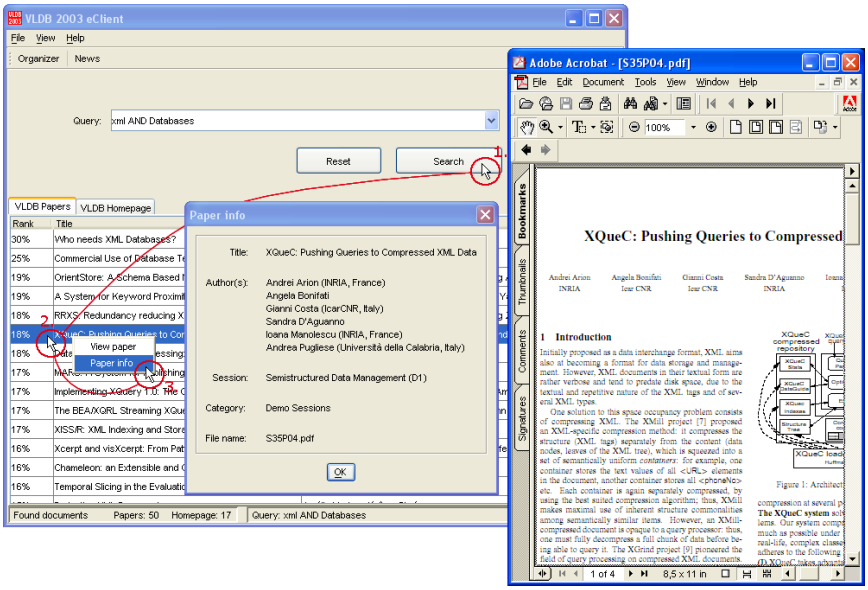

Abbildung 5.3: Volltextsuche und Level of Detail Konzept

Abbildung 5.3 zeigt das Suchpanel des *eClient* und die Umsetzung des Level of Detail Konzeptes. Die Liste der gefundenen Dokumente enthält zunächst nur wenige Informationen einschließlich des Dokumentennamens. Diese Ansicht entspricht der Granularitätsstufe “Label”, obwohl bereits einige Metadaten angezeigt werden. Über einen Dialog können weitere Metadaten abgerufen werden und bei einem Doppelklick auf den Listeneintrag wird das komplette Dokument geladen.

Eine Suche über Metadaten wurde bis jetzt nicht umgesetzt. Die notwendigen Daten werden aber bei der Indizierung der Dokumente erfasst, so dass eine solche Funktion in zukünftigen Version implementiert werden könnte.

Kapitel 6

Personalisierung

Eine digitale Bibliothek dient vor allem der Suche nach Dokumenten. Darüber hinaus sind aber weitere Dienste denkbar, die ein System aufwerten und für den Anwender attraktiver machen. Ein Nutzer möchte z.B. eine eigene Sicht auf die gespeicherten Daten erstellen, sich Notizen zu bestimmten Dokumenten machen oder informiert werden, falls sich Dokumente ändern.

Unter dem Begriff Personalisierung werden Konzepte zusammengefasst, die ein Bibliotheksystem an die speziellen Eigenschaften und Bedürfnisse eines Nutzers anpassen. Im Folgenden werden einige Konzepte, die bezüglich eines Konferenzszenarios interessant sind, kurz erläutert werden. Anschließend werden die im *eClient*-System umgesetzten Konzepte beschieben.

6.1 Konzepte

Nutzerprofile Grundlage für eine Personalisierungskomponente ist die Verwaltung von nutzerspezifischen Daten. Dies umfasst Angaben wie den Nutzernamen, die Email-Adresse oder die Interessengebiete eines Anwenders. Auf diese Daten können andere Dienste der Personalisierungskomponente aufbauen. Über die Email-Adresse kann ein Nutzer z.B. über Änderungen an Dokumenten benachrichtigt werden

Zur Erhebung der Daten sind zwei Varianten denkbar: Entweder der Nutzer gibt die Daten dem System bekannt, dies könnte z.B. am ersten Konferenztag bei der Registrierung erfolgen oder das System erfasst die Daten selbst, indem der Anwender bei der Benutzung des Systems beobachtet wird. Auch eine Kombination beider Verfahren ist möglich.

Eigene Sicht auf Daten Die Daten eines Bibliotheksystems sind nicht für jeden Nutzer gleich relevant. Bestimmte Nutzergruppen interessieren sich auf Grund ihrer spezifischen Interessen häufig nur für einen Teil der Daten. Es ist daher sinnvoll, den Nutzern die Möglichkeit zu bieten weniger relevante Daten auszublenden. Im Datenbankbereich ist dieses Konzept als Sicht bekannt.

In einem Konferenzsystem kann dieses Verfahren vor allem auf den Veranstaltungsplan angewendet werden. Ein Besucher sollte sich auf Grund seiner Interessen einen individuellen Plan erstellen können, der alle Sessions umfasst die ihn interessieren und die er besuchen möchte. Sind im Nutzerprofil die Interessen oder die Forschungsrichtung eines Konferenzbesuchers erfasst worden, ist es möglich, dass das System automatisch einen auf den Nutzer zugeschnittenen Veranstaltungsplan erstellt.

Annotation Oft ist es hilfreich sich während einer Präsentation Notizen zu machen, z.B. als Basis für im Anschluss an den Vortrag zu stellende Fragen. Dazu sollte es möglich sein, kurze unformatierte Texte zu erstellen, die einer Präsentation zu geordnet sind.

Prefetching In Abschnitt 3.2 wurde im Rahmen der verteilten Datenhaltung das Caching und Prefetching von Dokumenten erläutert. Beim Prefetching ging es darum Dokumente in den lokalen Cache der Client-Systeme zu laden, noch bevor der Nutzer die Dokumente anfordert. Für die Auswahl der im voraus zu cachenden Dokumente sind die im Rahmen der Personalisierung angelegten Nutzerprofile sehr gut geeignet. Auf Grund der angegebenen Nutzerinteressen oder des selbst erstellten Veranstaltungsplanes können entsprechende Dokumente ausgewählt werden, die dann offline verfügbar gemacht werden.

Weitere Konzepte zur Personalisierung und ausführlichere Erläuterungen der vorgestellten Techniken befinden sich beispielsweise in [Zei01].

6.2 Umsetzung

Einige der Konzepte wurden in die *eClient*-Applikation integriert.

Eigener Tagungsplan Die Applikation bietet die Möglichkeit, durch Auswahl einzelner Sessions einen eigenen Veranstaltungsplan zu erstellen. Alle nicht ausgewählten Sessions können anschließend ausgeblendet werden. Abbildung 6.1 zeigt den vollständigen zweiten Konferenztag der VLDB 2003 auf der linken Seite und einen individuellen Tag der nur ausgewählte Sessions enthält auf der rechten Seite.

Prefetching Über die Session-Auswahl und den so erstellten Tagungsplan wurde ein Prefetching-Mechanismus implementiert. Dabei wird davon ausgegangen, dass der Nutzer früher oder später auf die Dokumente der Präsentationen der ausgewählten Sessions zugreift. Wird eine Session markiert, werden die Dokumente der Präsentationen in eine Warteschlange gestellt und im Hintergrund in den Cache geladen. Greift der Nutzer schließlich auf die Dokumente zu, sind diese bereits lokal verfügbar und müssen nicht vom Server geladen werden.

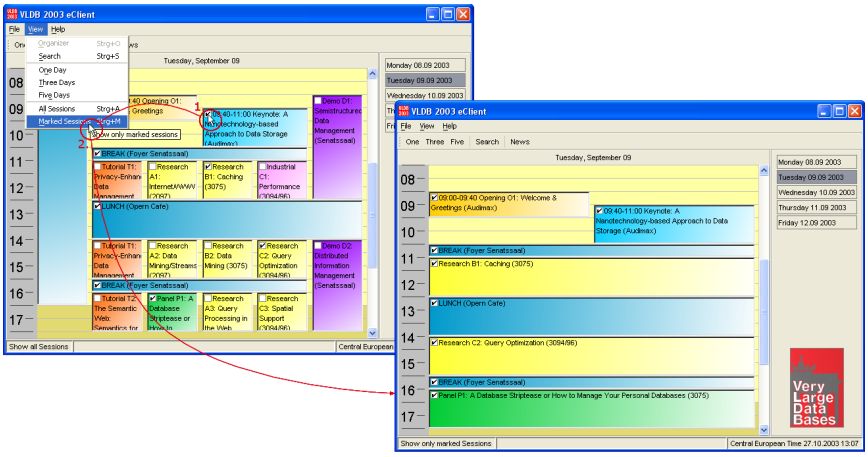

Abbildung 6.1: Erstellung eines eigenen Tagungsplans mit dem *eClient*

Reminder Als letzte Funktion, die im weitesten Sinne zu den Personalisierungskonzepten gezählt werden kann, wurde eine Erinnerungsfunktion implementiert. Darüber ist es möglich, sich an den Beginn einer ausgewählten Session erinnern zu lassen. Die Funktion beachtet dabei die Zeitverschiebung, welche zwischen dem Konferenzland und dem mobilen Gerät des Konferenzbesuchers auftreten kann. Der Reminder ist in Abbildung 6.2 dargestellt.

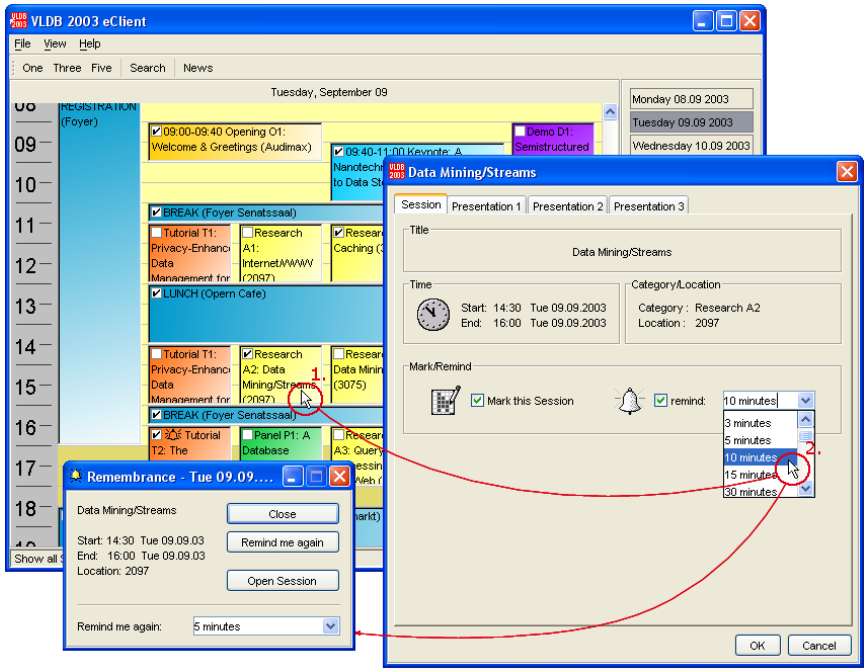

Abbildung 6.2: Die Reminder Funktion der *eClient*-Applikation

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel wird die vorliegende Studienarbeit zusammengefasst. Abschließend werden einige Erweiterungen vorgeschlagen, die in zukünftigen Arbeiten aufbauend auf diesem Projekt verwirklicht werden können.

7.1 Zusammenfassung

In dieser Arbeit wurde ein System entwickelt, welches Konferenzbesuchern den mobilen Zugriff auf die digitalen Inhalte einer Konferenzdomäne ermöglicht. Nach der Beschreibung des Anwendungsszenarios wurden Anforderungen erarbeitet, die ein solches System erfüllen muss. Ein wichtiges Ergebnis der Anforderungsanalyse, war die Notwendigkeit einer verteilten Datenhaltung, welche Datenzugriffe auch bei einer unterbrochenen Netzwerkverbindung ermöglicht und so eine "echte" Mobilität des Anwenders garantiert.

Anschließend wurde eine Systemarchitektur vorgestellt, welche die erarbeiteten Vorgaben umsetzt. Die Architektur wurde mit Hilfe der Java-Technolgie plattformunabhängig in dem *eClient*-System verwirklicht. Dabei wurde eine Datenhaltung entwickelt, welche die Daten auf drei Medien verteilt: den Datenservern, den Client-Cache und eine vom Konferenzveranstalter produzierte CD. Sowohl die Verteilung der Daten der einzelnen Datenklassen, als auch Strategien für den effizienten Zugriff auf die verteilten Daten, wurden ausführlich erläutert. Anschließend wurde mit dem JNLP-Client *Java Web Start* ein System zum Deployment und zur Wartung von Java-Applikationen vorgestellt und beschrieben, wie diese Technik in die Systemarchitektur des *eClient*-Systems integriert wurde.

Im fünften Kapitel wurde der Aufbau der *eClient*-Oberfläche dargestellt. Hier wurden Techniken wie das "Level of Detail"-Konzept erläutert und die verschiedenen Formen der Informationsgewinnung beschrieben, welche ein Client für digitale Bibliotheken unterstützen kann.

Abschließend wurden einige Konzepte zur Personalisierung von digitalen Bibliotheken eingeführt und die im *eClient*-System umgesetzten Techniken, wie das Dokumenten-Prefetching und das Erstellen eigener Tagungspläne, beschrieben.

7.2 Ausblick

In den einzelnen Kapiteln wurden bereits Vorschläge für die funktionale Erweiterung des entwickelten Systems gemacht, die zumeist aus Zeitgründen nicht mehr umgesetzt werden konnten. Die wichtigsten sollen hier hier überblicksartig dargestellt werden:

- *Metadaten Suche:* Sinnvoll wäre die Implementierung einer Metadatensuche mit Hilfe der *Lucene*-Indexstrukturen. Die hierfür notwendigen Daten werden bereits erfasst, so dass lediglich die *eClient*-Oberfläche erweitert werden müsste.
- *Annotationen:* Durch Annotationen könnte dem Anwender die Möglichkeit gegeben werden, Notizen zu Präsentationen zu erstellen, die aus kurzen unformatierten Texten bestehen.
- *Sessionvorschläge:* Im Rahmen der Nutzerprofile wäre es möglich die Forschungsgebiete und Interessen der Konferenzteilnehmer zu erfassen. Auf dieser Grundlage könnten vom System Vorschläge gemacht werden, welche Sessions ein Konferenzteilnehmer besuchen sollte.
- *Positionserfassung:* Des Weiteren kann darüber nachgedacht werden, die Position jedes Konferenzteilnehmers zu ermitteln und so situationsabhängige Daten proaktiv darzustellen. Dies würde allerdings eine entsprechende Infrastruktur zur Positionsbestimmung der Konferenzteilnehmer innerhalb des Konferenzgebäudes voraussetzen.

Die Konzeption und Entwicklung des *eClient*-Systems war von Anfang an auf eine Weiterführung im Rahmen des NUR-Projekts (Notebook Universität Rostock) der Universität Rostock ausgelegt. Innerhalb des Projektes soll das entwickelte System angepasst werden, um den Studenten der Universität einen mobilen Zugriff auf die Dokumente und Informationen des Vorlesungsbetriebs zu ermöglichen. Die Anwendungsdomäne ist der eines Konferenzszenarios dabei sehr ähnlich: Statt Sessions und Präsentationen wird der Anwender hier zwischen Vorlesungen der verschiedenen Fakultäten auswählen könnten. Dabei soll es ebenfalls möglich sein, persönliche Stundenpläne zu erstellen und auf Foliensätze und Skripte der Veranstaltungen zuzugreifen. Auf Grund des erheblich größeren Datenvolumens welches in diesem Fall zu verwalten ist, sind aber einige Änderungen in der Datenverwaltung notwendig. Dies betrifft beispielsweise den Veranstaltungsplan, in diesem Fall der Vorlesungsplan der Universität, der bisher in einer XML-Datei gespeichert wird. Diese Daten sollten in einem Datenbanksystem verwaltet werden, wie dies in Kapitel 4.4 bereits angedeutet wurde, da die Verwaltung großer Datenmengen in einer XML-Datei uneffizient ist.

Literaturverzeichnis

[AWT] Java GUI-Programme mit AWT und Swing.
http://www.ssw.uni-linz.ac.at/Teaching/Lectures/PSW2/2003/AWTundSwing.pdf.

[BAK+02] Suman Banerjee, Sulabh Agarwal, Kevin Kamel, Andrzej Kochut, Christopher Kommareddy, Tamer Nadeem, Pankaj Thakkar, Bao Trinh, Adel Youssef, Moustafa Youssef, Ronald L. Larsen, A. Udaya Shankar, and Ashok Agrawala. Rover: Scalable Location-Aware Computing. *IEEE Computer*, 35(10):46–53, 2002.

[Bau02] Gerald Bauer. Web Start Overview - Java's Comback On The Desktop, JUG Austria Talk, January 2002.
http://www.vamphq.com/talk/jug-jan-2002/slides.html.

[Ber02] David Berlind. Wer profitiert von C# - Microsoft oder die ganze Branche?, 2002.
http://web02.zdnet.de/enterprise/artikel/os/200210/2124632-p1-wc.html.

[Dib01] Jens Dibbern. Alternativen für die Entwicklung von grafischen Oberflächen in Java, 2001.
http://www.informatik.uni-bremen.de/ bohnebec/gi-rg/rg0412graphischeOberflaechen.ppt.

[Dol03] Sebastian Dolke. Einsatz von ER zum Entwurf von DTDs bzw. XML-Schema, Informatikseminar (Universität Rostock), Mai 2003.
http://wwwdb.informatik.uni-rostock.de/Lehre/Vorlesungen/hs_ws2002_2003/hs_dolke.ps.

[DSAF99] Anind K. Dey, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa. The Conference Assistant: Combining Context-Awareness with Wearable Computing. In *Proceedings of the 3rd International Symposium on Wearable Computers*, 1999.

[EF00] Albert Endres and Dieter W. Fellner. *Digitale Bibliotheken*. 2000.

[Eng03] Holm Engelbrecht. Gestaltung von Oberflächen - Die Quellen der Entwickler, Usability Seminar (Universität Rostock), 2003.
http://wwwswt.informatik.uni-rostock.de/deutsch/Mitarbeiter/michael/lehre/Usab_WS2002/Holm/.

[Fra02] Matthias Frank. Nutzen Digitaler Bibliotheken, Seminar der Universität der Bundeswehr München, März 2002.
http://www2-data.informatik.unibw-muenchen.de/Lectures/WT2002/DigiBib/.

[Haa01] Peter Haase. *Situationsgesteuerter mobiler Zugriff auf Digitale Bibliotheken*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, November 2001.

[Jol] J.M. Joller. Java Deployment oder Wie werden Java Applikationen ausgeliefert?
http://www.joller-voss.ch/ndkjava/notes/deployment/JavaDeployment.pdf.

[Jon99] Michael Jonas. *Ein Replikationsdienst für komplexe Dokumente.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, Juni 1999.

[Kur02] Matthias Kurth. UMTS und/oder WLAN, 2002.
http://www.regtp.de/aktuelles/02600/01/.

[Luc03] The Jakarta Lucene Project Homepage, 2003.
http://jakarta.apache.org/lucene/docs/index.html.

[Mar02] Björn Martin. *Beschleunigung von Java-Oberflächen mit OpenGL*. Bachelorarbeit, Fachhochschule Karlsruhe, Fachbereich Informatik, 2002.
http://www.stz-ida.de/html/forschung/ba_martin.html.de.

[New03] Heise Newsletter. Klapp-Handy und mobile Surfstation, Oktober 2003.
http://www.heise.de/newsticker/data/rop-28.10.03-000/.

[nib] The Nibble Location System.
http://mmsl.cs.ucla.edu/nibble/.

[Pae96] Andreas Paepcke. Digital Libraries: Searching Is Not Enough, Mai 1996.
http://www.dlib.org/dlib/may96/stanford/05paepcke.html.

[Rot02a] Jörg Roth. *Mobile Computing - Grundlagen, Technik, Konzepte*. dpunkt-Verlag, Mai 2002.

[Rot02b] Jörg Roth. Script der Vorlesung Mobile Computing, Universitt Dortmund, 2002.
http://dreamteam.fernuni-hagen.de/vorl_mobilecomputing/vorlesung.html.

[SCG+02] Holger Schwichtenberg, Sven Conrad, Thomas Gartner, Peter Krause, and Oliver Scheer. *Microsoft ASP.NET. Das Entwicklerbuch*. Microsoft Press, 2002.

[Sch02] Alexander Schill. Script der Vorlesung Mobile Kommunikation und Mobile Computing, TU Dresden, 2002.
http://www.rn.inf.tu-dresden.de.

[Uni] Lancaster University. Java Support on Pocket PC.
http://www.comp.lancs.ac.uk/computing/users/fittond/ppcjava.html.

[Wat98] Donald J. Waters. What are Digital Libraries?, CLIR issues, Number 4, July/August 1998.
http://www.clir.org/pubs/issues/issues04.html.

[WSG] Java Web Start Developer's Guide Version 1.2.
http://java.sun.com/products/javawebstart.

[WSH] Java Web Start Architekture Website.
http://java.sun.com/products/javawebstart/architecture.html.

[Zei01] Andre Zeitz. *Personalisierungskonzepte für Digitale Bibliotheken*. Diplomarbeit, Universität Rostock, Fachbereich Informatik, November 2001.

Abbildungsverzeichnis

Selbständigkeitserklärung

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

Neubrandenburg, den 15.01.2004

_____

Sebastian Dolke