Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Every thing has online content

Diplomarbeit

Entwurf und Implementierung des ETHOC-Systems

Nikolaos Kaintantzis
kaintantzis@iaeth.ch

Betreut von: Jürgen Bohn und Michael Rohs
Unter der Aufsicht von Prof. Dr. Friedemann Mattern
Institut für Informationssysteme, Fachgruppe Verteilte Systeme

Abgabe: 4. März 2002

Inhaltsverzeichnis

Zusammenfassung **xi**

1 Einleitung **1**
1.1 Ubiquitous Computing 1
1.2 Ziel dieser Arbeit 2
1.3 Kapitelübersicht 3

2 Motivation **5**
2.1 Beispiele für online erweiterte Dokumente 5
2.1.1 Veranstaltungshinweise 6
2.1.2 Ankündigungen 7
2.1.3 Unterrichtsbezogene Dokumente 9
2.1.4 Weitere Dokumente 10
2.2 Verallgemeinerung auf beliebige Gegenstände 10

3 Technologien und Plattformen **13**
3.1 Identifikationstechnologien 13
3.1.1 Barcodes 14
3.1.2 Weitere Tagging-Systeme 16
3.2 Barcode-Leser 17
3.3 Klienten-Plattformen 18
3.3.1 PC 18
3.3.2 Laptop 18
3.3.3 PDA 18
3.3.4 Mobiltelefon 19

4 Systemanalyse und -Anforderungen **21**
4.1 Benutzungsszenarien 21
4.1.1 Filmveranstaltung und drahtlose Kommunikation 21
4.1.2 Vorlesung und Studenten-Raum 22
4.1.3 Aktionen auf Kleinstrechner 22
4.1.4 Annonce und Mobiltelefon 23
4.2 Autorenunterstützung 23
4.2.1 Erstellungsprozess 23
4.2.2 Änderungsprozess 24
4.2.3 Autorbezogene Systemkomponenten 24
4.3 Benutzerunterstützung 26
4.3.1 Vermittlungsprozess 26

4.3.2 Benutzerbezogene Systemkomponenten 27
4.4 Zusammenfassung 28

5 Konzepte für das ETHOC-System 31
5.1 Barcode-Erstellung 31
5.2 Objekt-ID 32
5.2.1 Hashwert als ID 32
5.2.2 Globaler Zähler als ID 32
5.2.3 Tag-URI als ID 32
5.2.4 ETHOC-Tag als ID 33
5.3 Varianten im Erstellungsprozess 34
5.3.1 Webservice 34
5.3.2 Logischer lokaler Drucker 37
5.3.3 Logischer Online-Drucker 39
5.3.4 Varianten-Evaluation 41
5.4 Varianten im Vermittlungsprozess 43
5.4.1 Online ID-Manager 43
5.4.2 Variante ID-Manager mit lokaler Komponente 45
5.4.3 Variantenvergleich 47
5.5 Meta-Informationen 47
5.5.1 Aktionen 48
5.5.2 Module 48
5.6 Zusammenfassung 49

6 Design des ETHOC-Systems 51
6.1 XML-Strukturen und -Verarbeitung 51
6.2 Barcode 52
6.3 ETHOC-Portal 52
6.4 Autoren-Portal 53
6.4.1 Einstiegsseite 53
6.4.2 Objekt-Erstellung und -Änderung 54
6.4.3 Abstrakte Servlets 56
6.4.4 Servlet-Hierarchien 57
6.5 Datenbank 58
6.5.1 Zugriff auf die Datenbank 59
6.6 Benutzerseite 60
6.6.1 Lokale Komponenten 61
6.6.2 Online-Komponenten 62

7 Implementierung 65
7.1 XML-Strukturen 65
7.1.1 Virtuelle Gegenstücke 65
7.1.2 Modulkonfiguration 66
7.1.3 Autoren-Werkzeuge 66
7.2 Autorenportal 67
7.2.1 Einstiegsseite 67
7.2.2 Autoren-Identifikation 67
7.2.3 Virtuelle Gegenstücke erstellen und ändern 67
7.2.4 Datei-Upload 68
7.2.5 Barcode-Erzeugung 68

7.3 Module . . . 68
7.4 Email-Verkehr . . . 69
7.5 Benutzerportal . . . 70
7.6 ETHOC-Browser . . . 71
7.7 Datenbank . . . 72
7.8 Aspekte der Verteiltheit . . . 74
7.9 Sicherheitsaspekte . . . 75
7.9.1 Ausgezeichnete Benutzerrechte . . . 75
7.9.2 Datenübertragung . . . 75
7.9.3 Passwörter . . . 76

8 Erweiterung des ETHOC-Systems 79
8.1 Neue Module einbinden . . . 79
8.2 Erweiterung auf Gegenstände . . . 80
8.3 Neue ETHOC-Typen . . . 81
8.3.1 Autorenseitig . . . 81
8.3.2 Benutzerseitig . . . 82
8.4 Einbinden neuer Barcode-Leser . . . 82
8.4.1 Serverseitige Anbindung . . . 82
8.4.2 Anbindung im ETHOC-Browser . . . 83

9 Diskussion und Ausblick 85
9.1 Erfüllung der Aufgabenstellung . . . 85
9.2 Bewertung der Arbeit . . . 86
9.3 Verifikation der Benutzungsszenarien . . . 86
9.3.1 Filmveranstaltung und drahtlose Kommunikation . . . 86
9.3.2 Vorlesung und Studenten-Raum . . . 87
9.3.3 Aktionen auf Kleinstrechner . . . 87
9.3.4 Annonce und Mobiltelefon . . . 87
9.4 Vorschläge für weiterführende Arbeiten . . . 88
9.4.1 Modifizierende Arbeiten . . . 88
9.4.2 Erweiternde Arbeiten . . . 88

A Aufgabenstellung 91

B Glossar 95

C Eingesetzte Softwareprodukte 97
C.1 Windows 2000 . . . 97
C.2 JAVA . . . 97
C.3 Jakarta Tomcat . . . 97
C.4 XALAN . . . 98
C.5 XERCES . . . 98
C.6 MySQL . . . 98
C.7 MM.MySQL . . . 98
C.8 COS . . . 99
C.9 SUNs JPEG-API . . . 99
C.10 Acme Labs GIFEncoder . . . 99
C.11 SmtpClient . . . 100

D XML-Definitionen **101**
D.1 doc.dtd 101
D.2 modules.dtd 103
D.3 authoringtools.dtd 103

Abbildungsverzeichnis

3.1 Beispiele zweidimensionaler Barcodes 14
3.2 UPC Version A und EAN13 15
3.3 Aufbau eines Code 128 Barcodes 16
3.4 Beispiel eines Code 39 Barcodes 16

4.1 Prinzip des Erstellungsprozesses 23
4.2 Zusammenspiel der autorbezogenen Systemkomponenten 24
4.3 Prinzip des Vermittlungsprozesses 26
4.4 Zusammenspiel der benutzerbezogenen Systemkomponenten . . . 27

5.1 Code 39 vs. Code 128 31
5.2 Variante Webservice 34
5.3 Variante logischer lokaler Drucker 37
5.4 Variante logischer Online-Drucker 39
5.5 Variante Online-ID-Manager 43
5.6 ID-Manager mit lokaler Komponente 45

6.1 Besipiel eines ETHOC-Barcodes 52
6.2 Einstieg ins Autorenportal 53
6.3 Ablauf der Objekt-Erstellung und -Änderung 54
6.4 Methoden des abstrakten ETHOC-Servlets 56
6.5 Methoden des abstrakten ETHOC-XML-Servlets 56
6.6 Methoden des abstrakten ETHOC-Service-Servlets 57
6.7 Servlet-Hierarchie im Autorenbereich 58
6.8 Zugriff auf Informationen in der Datenbank 59
6.9 Benutzungsmöglichkeiten des ETHOC-Systems 60
6.10 Umgang mit der zentralen Historie 61

7.1 Klassen und Navigation im Benutzerportal 69
7.2 Hauptklassen des ETHOC-Browsers 71

Tabellenverzeichnis

3.1 Vergleich Barcode vs. RFID-Marken 13

5.1 Evaluation Webservice 41
5.2 Evaluation lokaler Druckertreiber 41
5.3 Evaluation Online-Drucker 42
5.4 Meta-Informationen eines Dokumentes 48
5.5 Aktionen eines Dokumentes 49

7.1 Definition der MySQL-Tabelle “IdCoord” 73

Zusammenfassung

Abstract

In a ubiquitous world physical objects (like paper documents) possess an electronic representation, so-called virtual counterparts, which supply additional functionality and resources. Various mobile devices allow an interaction with such virtual counterparts.

The subject of this work is the design and implementation of a basic service for the linkage of physical documents with on-line resources and functions. Bar codes and mobile bar code readers are used as the linking technology to access virtual resources as well as additional functions of printed documents. The ETHOC system makes an infrastructure available, which supports the production, administration and intermediation of on-line resources. To authors of documents, the system supplies an XML and Web-based portal. On the user side different entry possibilities exist to access virtually extended documents: the ETHOC browser for Java enabled devices, e.g. PDAs or laptops, a WML interface for mobile telephones, as well as a Web portal for access over conventional Web browser. The information is displayed according to the capabilities of the devices used. The personal history of earlier accesses is also available. The user can access the respective on-line resources and/or makes use of additional functions of the document, for example the automatic creation of calendar entries. Furthermore, the ETHOC browser allows a local caching of on-line resources, so that data remain available in an off-line operation later on.

Special attention in the design of the ETHOC System has been paid to flexibility and extensibility. The function range on client and server side can be simply extended by modules and new object types. In particular, a further development of the ETHOC system is intended to support any objects, so that the vision "every thing has on-line content" will come true!

Zusammenfassung

In einer ubiqitären Welt besitzen physische Objekte (wie z.B. Papierdokumente) elektronische Repräsentationen, sogenannte virtuelle Gegenstücke, die weitergehende Funktionalität und Ressourcen bereitstellen. Eine Interaktion mit solchen virtuellen Gegenstücken wird durch verschiedenste mobile Geräte ermöglicht.

Gegenstand dieser Arbeit ist die Konzeption und praktische Verwirklichung eines Basisdienstes zur Verknüpfung physischer Dokumente mit Online-Ressourcen und -Funktionen. Für den Zugriff auf virtuelle Ressourcen und Zusatzfunktionen gedruckter Dokumente werden Barcodes und mobile Barcode-Lesegeräte als Verknüpfungstechnologie verwendet. Mit dem ETHOC-System wird darüber hinaus eine Infrastruktur bereitgestellt, welche die Erzeugung, Verwaltung und Vermittlung von Online-Ressourcen unterstützt. Das System stellt den Autoren von Dokumenten dazu ein XML- und Web-basiertes Autorenportal bereit. Auf der Benutzerseite existieren verschiedene Zugangsmöglichkeiten zu virtuell erweiterten Dokumenten: der ETHOC-Browser für Java-fähige Geräte, wie z.B. PDAs oder Laptops, ein WML-Interface für Mobiltelefone, sowie ein Web-Portal für den Zugriff über herkömmliche Web-Browser. Dabei werden die Informationen angepasst an die Leistungsmerkmale der jeweiligen Geräte dargestellt. Die persönliche Historie früherer Zugriffe kann eingesehen und auf die jeweiligen Online-Ressourcen zugegriffen bzw. zusätzliche Funktionen zum Dokument aufgerufen werden, wie beispielsweise die automatische Erstellung von Kalendereinträgen. Der ETHOC-Browser ermöglicht desweiteren ein lokales Caching von Online-Ressourcen, so dass einmal eingelesene Daten auch in einem späteren Offline-Betrieb verfügbar bleiben.

Bei der Konzeption des ETHOC-System wurde besonderes Augenmerk auf Flexibilität und Erweiterbarkeit gelegt, damit sich der Funktionsumfang auf Client- und Serverseite mit Hilfe von Modulen und neuen Objekt-Typen nachträglich einfach erweitern lässt. Insbesondere ist eine Weiterentwicklung des ETHOC-Systems zur Unterstützung beliebiger Objekte vorgesehen, so dass sich die Vision “Every thing has online content” erfülle.

Kapitel 1

Einleitung

ETHOC steht für "every thing has online content". Dieses Kapitel motiviert die Idee dieser Arbeit und ordnet sie in bestehende Konzepte ein. Es zeigt auch auf, wie das ETHOC-System in den Alltag einbezogen werden kann.

1.1 Ubiquitous Computing

Durch den technischen Fortschritt werden Prozessoren immer kleiner und kostengünstiger. Die Miniaturisierung wird in naher Zukunft dazu führen, dass sie industriell in Alltagsgegenstände eingebaut werden können. Erweitert mit Sensoren und drahtlosen Kommunikationsmöglichkeiten werden diese Prozessoren untereinander Informationen austauschen. Die Computer sind allgegenwärtig – ubiquitär. Mattern schreibt in seinem Artikel [1]: "Von einem technikzentrierten Standpunkt kann 'ubiquitous computing' als die Aussicht, die restlichen Dinge der Welt ans Internet anzuschliessen, bezeichnet werden".

Vergleich man mit früher, so hat das Netzwerk an Wert gewonnen. So werden heute Computer gekauft, um ans Internet angeschlossen zu werden. Oder wie Mattern in seinem Artikel [2] schreibt: "Es ist nicht mehr der PC, der im Mittelpunkt steht und an den man die Netzperipherie anschliesst, sondern das Netz als solches hat eine unabhängige, dauerhafte Existenz angenommen."

Dinge können also mittels in ihnen versteckten Prozessoren Daten verarbeiten und miteinander und dem Internet kommunizieren; sie werden smart. Smarte Gegenstände verlangen im Gegensatz zu den komplexen PCs dem Benutzer nicht viel Aufmerksamkeit ab. Sie erledigen ihre Aufgabe (z.B. das Bewässern des Rasens unter Zuhilfenahme von Feuchtigkeitssensoren im Boden und Wetterberichten aus dem Internet) und unterstützen den Benutzer (z.B. indem bei absehbarem Regenrisiko dem Benutzer mitgeteilt wird, den Schirm mitzunehmen). Sie können sich ihren Aufenthaltsort und nahe Gegenstände merken; kennen also ihre Geschichte. Dadurch ist es möglich, sich z.B. beim Schreibtisch zu erkundigen, wo die aktuellen Aufenthaltstorte jener Bücher sind, die am Tag zuvor auf ihm lagen. Sie erkennen ausserdem ihre Benutzer. Dadurch kann auf Präferenzen implizit zurückgegriffen werden, statt sie explizit zu erfragen. So könnte der Kaffee-Automat automatisch die Zuckermenge ermitteln, die jemand

für gewöhnlich im Kaffee wünscht oder das Auto die Sitz- und Spiegeleinstellungen automatisch an die letzten Einstellungen des Fahrers anpassen. Durch die Kooperation mit anderen Objekten und Diensten erzeugen die Objekte einen Mehrwert, indem sie ihre eigene Arbeit besser automatisieren oder zusätzliche Information anbieten.

Doch wie können Dinge so viel Informationen behalten und wie können sie Informationen aus dem Web aufbereiten? Mehr und mehr Alltagsobjekte erhalten eine Repräsentation in einer Datenbank oder im Internet. Dies stellen auch Langheinrich et al. in [3] fest. Die Objekte besitzen also virtuelle Gegenstücke oder virtuelle Proxies. Nicht das Objekt selber, sondern sein virtuelles Gegenstück verwaltet die Daten und Beziehungen zu anderen Objekten. Das Objekt realisiert über Sensoren Änderungen in seiner Umgebung und teilt sie seinem virtuellen Gegenstück mit.

Das eindeutige Identifizieren ist die Minimalvoraussetzung für die Verbindung zwischen physischen "Objekten" und virtuellen "Gegenstücken". Erhalten Alltagsgegenstände eine elektronische Identität, können diese mit ihrem virtuellen Gegenstück verknüpft werden. Bohn und Rohs bezeichnen in [4] einen solchen Vermittler als eine symbolische Lupe. Wie die optische Lupe ermöglich die symbolische Lupe das nähere Betrachten eines Details. Mit dem Unterschied, dass hier die Details mit blossem Auge nicht erkennbar sind, sondern nur mit Hilfe der symbolische Lupe. Die symbolische Lupe ermöglicht den Eintritt in eine virtuelle Welt, wie z.B. dem Internet, ausgehend von einem physischen Objekt. Sie nimmt die ID auf und kontaktiert das virtuelle Gegenstück.

1.2 Ziel dieser Arbeit

Ziel dieser Arbeit ist die Zuordnung von Online-Ressourcen zu gedruckten Dokumenten, sowie die Untersuchung von Interaktionsmöglichkeiten mir diesen Ressourcen, ausgehend vom physischen Dokument.

Dokumente erhalten also ein virtuelles Gegenstück. Der Autor wird beim Erstellen und Verwalten des virtuellen Gegenstückes durch die zu schaffende Infrastruktur unterstützt. Der Benutzer erhält durch die Vermittlung der Infrastruktur Zugriff auf das virtuelle Gegenstück und somit einen Mehrwert, da das virtuelle Gegenstück dem Leser des Dokumentes ergänzend digitale Informationen und Aktionen anbietet. Es können Aktionen angeboten werden, die auf dem physischen Dokument nicht möglich sind, z.B. das Nachführen der Agenda des Benutzers durch das virtuelle Gegenstück eines Plakates, welches eine Veranstaltung ankündigt. Dabei gilt es, auf unterschiedliche Plattformen und Benutzungsszenarien einzugehen. Detailliertere Szenarien werden im Kapitel 4.1 besprochen.

Die Dokumente selbst müssen nicht smart sein. D.h. sie müssen ihre Umwelt nicht wahrnehmen können. Sie sollen lediglich eine Schnittstelle zum Internet erhalten. Dadurch können die virtuellen Gegenstücke mit den Geräten des Benutzers kommunizieren. Dies geschieht dann, wenn jemand die elektronische ID des Dokumentes aufsammelt und im Internet das virtuelle Gegenstück kontaktiert.

Jedes Objekt existiert einmalig und hat eine eigene elektronische ID. Dies beschreiben auch Bohn und Rohs in [4]. Dokumente aber können vervielfältigt oder mehrfach gedruckt werden, was durchaus wünschenswert sein kann. Die Kopien der Dokumente sollen denselben Einstiegspunkt ins Internet, also dasselbe virtuelle Dokument haben, wie das Original. Die IDs der Dokumente haben hier eine ähnliche Funktionalität wie die Produkt-Kodes im Laden, welche nicht die individuellen Produkte, sondern Produkttypen bezeichnen.

Die Aufgabenstellung (siehe Anhang A) verlangt explizit die Benutzung von Barcodes. Dieser Entscheid ist für dieses Projekt angemessen, denn die ID kann somit direkt in die Druckvorlage einfliessen. Zur Anbringung der ID wird also keine zusätzliche Infrastruktur benötigt, sie wird elektronisch durchgeführt. Die Vervielfältigung von Dokumenten mit Barcodes ist ebenfalls ohne zusätzlichen Aufwand resp. ohne Verlust der ID möglich. Würde man z.B. RFID-Tags den Barcodes vorziehen, so wären sicherlich einige weitergehende Szenarien denkbar (zum Beispiel könnte ermittelt werden, wieviele Leute vorbeilaufen und wieviele einen Aushang betrachten), doch wären der Aufwand und die Materialkosten für die Autoren höher. Wenn zum Beispiel ein Professor sein Vorlesungsskript mit einem Tag versehen will, so müsste er (oder seine Assistenten) einige hundert RFID-Tags anbringen. Jedes einzelne Tag müsste programmiert werden. Dieser Aufwand ist einzig durch den Kauf zusätzlicher Geräte reduzierbar. Denkbar wäre zum Beispiel eine Art Frankiermaschine, welche die Tags anbringt und gleich programmiert. Ein weiteres Problem der RFID-Tags ist das exakte Ausmachen von Dokumenten. Stünde der Benutzer vor einer Plakatwand, könnten plötzlich alle Dokumente sich angesprochen fühlen und ihre ID zum Leser der Benutzers schicken. Ein Vergleich der Technologien Barcode und RFID-Tags ist im Kapitel 3 zu finden.

1.3 Kapitelübersicht

Im weiteren ist der Bericht wie folgt gegliedert: Kapitel 2 motiviert den Einsatz von ETHOC, indem ausführliche Anwendungsbeispiele aufzeigt werden. Diese Beispiele bringen erste Anforderungen zu Tage, die in der Konzeption berücksichtigt werden. Die Technologien, die heute zur automatischen Identifikation zur Verfügung stehen, werden in Kapitel 3 beschrieben und verglichen. Des weiteren wird auf die Kombinationsmöglichkeiten dieser Technologien mit Klienten-Plattformen eingegangen. Kapitel 4 beschreibt vier typische Einsatzszenarien, die durch das ETHOC-System untertützt werden sollen. Diese Szenarien implizieren Anforderungen an das System, welche analysiert und Systemkomponenten zugeordnet werden. In Kapitel 5 werden verschiedene Varianten zur Realisierung dieser Komponenten vorgestellt und bewertet. Die ausgewählte Variante wird in Kapitel 6 ausgearbeitet. Details der Implementierung folgen in Kapitel 7. Kapitel 8 zeigt auf, wie das ETHOC-System durch neue Module und Typen nachträglich erweitert werden kann. Abschliessend wird in Kapitel 9 ein Fazit gezogen, und es werden Vorschläge für weiterführende Arbeiten vorgestellt. Im Anhang finden sich die Aufgabenstellung, ein Glossar, eine Übersicht der eingesetzten Softwareprodukte mit Installationshinweisen, sowie die Quelltexte der im ETHOC-System verwendeten XML-Definitionen.

Kapitel 2

Motivation

In diesem Kapitel werden Beispiele für mögliche Erweiterungen durch Online-Ressourcen vorgestellt und somit der Einsatz von ETHOC motiviert. Dieses Kapitel soll den Leser inspirieren, eigene Ideen oder Einsatzgebiete zu entwickeln.

2.1 Beispiele für online erweiterte Dokumente

Dieser Abschnitt illustriert wünschbare Aktionen für mögliche Dokument-Typen. (Alle möglichen Aktionen unabhängig vom Dokumententyp sind im Abschnitt 5.5.1 zusammengefasst.) Natürlich ist es möglich, zu jedem Dokument als virtuelles Gegenstück nur einen Link auf eine Internetseite vorzusehen, auf welcher ein HTML-Designer im Auftrag des Autors alle relevanten Informationen zum Anschauen und Runterladen anbietet. Dieses Projekt will aber Autoren ohne HTML-Kenntnisse oder ohne Zeit, einen Internet-Auftritt zu organisieren, helfen, ein virtuelles Gegenstück ihres Dokumentes in kurzer Zeit zu erstellen.

Die Basis-Funktionalität von virtuellen Dokumenten wird im Kern des Systems verwirklicht. Es wird aber eine Schnittstelle geben, um weitere Funktionen und Dienste, sogenannte Module, in ETHOC einzubauen. Module verwirklichen weitere Dienste, die sich Autoren für virtuelle Dokumente wünschen, die aber im ETHOC-System nicht verwirklicht worden sind, sei es, um die grundsätzliche Benutzerführung einfach zu halten oder weil der Zeitrahmen dieser Diplomarbeit es nicht erlaubt hätte. Dank Modulen sind Erweiterungen ohne Anpassung des bestehenden Kodes möglich. Es können beliebige Dienste und Funktionen eingebaut werden, die der Autor bei Bedarf auswählen kann. Eine zu klärende Frage wird sein, ob alle folgenden Dienste im ETHOC-System als Module angeboten werden sollen oder ob sie Links zu bestehenden Diensten sein sollen, welche der Autor selbst finden und einrichten muss. Zum Beispiel soll die Frage beantwortet werden, ob das ETHOC-System Diskussionsforen anbieten oder lediglich auf Foren im Internet verweisen soll.

2.1.1 Veranstaltungshinweise

Veranstaltungen finden an einem bestimmten Ort während einer gewissen Zeitperiode statt. Würden Veranstaltungshinweise mit einem Barcode versehen, so könnte das virtuelle Gegenstück mit folgenden Aktionen einen Komfort bieten:

- Termin automatisch in den Kalender eintragen
- Dokument herunterladen
- Wegbeschreibung oder Gebäudeplan anzeigen
- Diverse Links zum Thema anzeigen
- Themenrelevante Dokumente herunterladen

Filmvorführung

An der ETH werden von Studenten-Vereinen (z.B.: VIS oder SOS-ETH) Filmabende organisiert. Die Studenten werden mit Plakaten auf diese Filme aufmerksam gemacht. Eine virtuelles Gegenstück könnte folgende Aktionen anbieten:

- Filmkritik anzeigen
- Plakat herunterladen
- Trailer anzeigen
- Link zum Reservationssystem des Veranstalters anzeigen
- Link zur Homepage des Films anzeigen

Vortrag

An der ETH sowie an anderen Hochschulen werden regelmässig Vorträge – sei es durch Angehörige der Hochschule oder durch Vertreter aus der Wirtschaft – gehalten. Eine virtuelles Gegenstück könnte folgende Aktionen anbieten:

- Porträt der Referenten herunterladen
- Folien herunterladen
- Abstrakt herunterladen
- Link zum Anmeldeformular, falls Platzzahl beschränkt oder Vortrag kostenpflichtig
- Link zu relevanten Publikationen anzeigen
- Link zu Feedbackformular anzeigen

Einführungsvorlesung

Neue Professoren halten in der Regel kurz nach ihrem Eintritt eine Einführungsvorlesung, in welcher sie ihr Forschungsgebiet vorstellen. Zusätzlich zu einem normalen Vortrag bieten sich folgende Aktionen für ein virtuelles Gegenstück an.

- Lebenslauf des Professors herunterladen
- Link zur Instituts-Homepage des Professors anzeigen
- Link zur Homepage des Professors anzeigen

Seminar

Eine Ankündigung für ein Seminar besteht in der Regel aus einer Reihe von Terminen, in welchen pro Termin ein Thema behandelt wird. Der erste Termin ist meistens eine Vororientierung. In der Regel wird das Leitthema vorgestellt und Beispielthemen für Vorträge geschildert. Eine virtuelles Gegenstück könnte einen Link zum Registrierungsformular (inkl. Themenvergabe) als Aktion anbieten.

2.1.2 Ankündigungen

Ankündigungen informieren über Geschehnisse rund um den Alltag. Virtuelle Gegenstücke zu Ankündigungen können folgende Aktionen beinhalten:

- Kontakt zum Autor
- Kontakt zur zuständigen Person (z.B. Sekretariat)
- Dokument herunterladen

Semester- und Diplomarbeiten

Ein grosser Teil der Aushänge im Informatik-Gebäude der ETH sind Beschreibungen von Semester- und Diplomarbeiten. Sie skizzieren im Groben die Aufgabenstellung, verweisen auf andere Arbeiten und nennen Kontaktadressen für weitere Informationen. Eine virtuelles Gegenstück könnte Links zu Vorarbeiten (evtl. Demoversionen von Vorarbeiten) und Links zu anderen Semester- und Diplomarbeiten des selben Instituts anbieten.

Bestandenen-Liste

Viele Departemente der ETH hängen nach den Prüfungen Listen mit den Namen oder Leginummern der Studierenden aus, welche die Prüfungen bestanden haben. Eine Notenübersicht der einzelne Prüfungen kann beim Studiensekretariat abgeholt oder per Post zugestellt werden. Zudem hat jeder Student das Recht, seine Prüfungen anzuschauen und die Korrektur mit einem Assistenten

zu besprechen. Wo und wann diese Besprechungen stattfinden, wird später nochmals separat angeschlagen. Eine virtuelles Gegenstück könnte folgende Aktionen anbieten:

- Nach der Notenkonferenz die Verfügung mit den Noten herunterladen
- Informationen downloaden, wo welche Prüfungen einsehbar sind
- Termine und Beschreibung für weiteres Vorgehen herunterladen

Aus Datenschutzgründen sollte die Bestandenenliste auf keinen Fall heruntergeladen werden können. Das virtuelle Gegenstück darf also in diesem Szanario nur die Aktionen beinhalten und nicht die Online-Kopie des ausgehängten Dokuments.

Aktualisierung des Vorlesungsverzeichnisses

Das Vorlesungsverzeichnis wird an der ETH so früh gedruckt, dass Vorlesungen von neuen Dozenten nicht aufgeführt sind und erst nach dem Druck gemeldet werden können. Dadurch wird jedes Semester eine Ankündigung mit den nicht erfassten Vorlesungen ausgehängt. Würde dieser Aushang mit einem Barcode versehen, könnte dieser auf eine stets aktualisierte Übersichtseite aller Veranstaltungen verweisen.
Wenn mehrere Barcodes pro Dokument erlaubt sind, könnte jede Veranstaltung mit einem Barcode versehen werden, der auf die Homepage der Veranstaltungen führt.

Wohnungsvermittlung

Eine virtuelles Gegenstück könnte folgende Aktionen anbieten:

- Bilder der Wohnung und Umgebung anzeigen
- Grundriss der Wohnung anzeigen
- Anzeige des Wohnstandortes im Stadtplan
- Kontakt zum Vermieter

Probandensuche

Diplomanden im Sport- oder Psychologiebereich suchen für Studien Probanden, die gewisse Voraussetzungen erfüllen müssen. Durch einen Online-Fragebogen könnte das virtuelles Gegenstück helfen, ungeeignete Probanden auszuschliessen.

2.1.3 Unterrichtsbezogene Dokumente

Vorlesungsskripte

Jedes Vorlesungsskript könnte mit einem Barcode versehen werden, der auf die elektronische Version verweist. Alternativ könnte für jedes Skript einer Vorlesungsstunde der gleiche Barcode verwendet werden. Dann müsste das ETHOC-System eine Historie der Online-Kopien verwalten können, damit ältere Kapiteln des Skripts über ihren aufgedruckten Barcode noch erreichbar sind. Wird das Skript um Kapitel ergänzt, können die alten Versionen gelöscht werden. Die Wahl liegt beim Autor.

Das aktuelle Vorlesungsskript könnte auf einzelne multimediale Dokumente der aktuellen Vorlesung verweisen. So könnte während dem Unterricht jeder Student die Folien runterladen und sein persönliches Exemplar ergänzen oder allfällige Fehler korrigieren. Mathematische Vorlesung könnte als Beilage eine Maple-Datei enthalten und dadurch Studierende animieren, eigene Ideen und Lösungsvorschläge auszuprobieren.

Übungen

Praktische Übungen müssen in einer bestimmten Software-Umgebung absolviert werden. Hierfür müssen bestimmte Programme und Bibliotheken installiert sein. In den Aufgabenblättern steht, welche Befehle eingegeben werden müssen, damit die Umgebung richtig konfiguriert wird. Zu Programmieraufgaben gibt es häufig vorgegebenen Kode, der ergänzt werden muss. Zu einem bestimmten Punkt erscheint dann auch die Musterlösung. Eine virtuelles Gegenstück könnte folgende Aktionen anbieten:

- Download der Konfigurationsskripte
- Download bestehender Kode-Fragmente
- Verknüpfung mit der Musterlösung, die erst ab einem gewissen Zeitpunkt zugänglich ist
- Link zur Testatübersicht für einen Studenten

Letzterer Punkt ist ein Beispiel für eine benutzerabhängige Aktion. Alle Studenten haben dasselbe Übungsblatt vor sich, sprich, denselben Barcode. Dennoch könnte es möglich sein, dass jeder Student nur seine Testate sieht. Das Authentifizieren könnte dadurch geschehen, dass beim Auflösen der Barcode-ID benutzerspezifische oder Barcode-Leser-spezifische Daten, wie z.B. Seriennummer, geschickt werden. Sie könnte aber auch via einen Login auf der Testatsübersichtsseite erfolgen.

Einschreibebogen

Vor jedem Semester erhält jeder Student seinen Einschreibebogen, den er zu kontrollieren, gegebenenfalls zu korrigieren und ihn an die Schulleitung zurückzusenden hat. Ein Barcode auf dem Einschreibebogen könnte den Studenten aufs Online-Verwaltungssystem der ETH führen.

2.1.4 Weitere Dokumente

Abstimmung

Wenn mehrere Barcodes pro Dokument unterstützt werden, könnten Umfragen direkt über das Auflösen eines Barcodes ausgeführt werden. Das ETHOC-System müsste einerseits sicherstellen, dass kein Benutzer mehrmals teilnehmen kann andererseits aber auch, dass keine Rückschlüsse zur Person gezogen werden können, falls die Abstimmung anonym verläuft. Ebenfalls erkannt werden, müsste das Erfassthaben von mehreren Kodes, die zur gleichen Abstimmung gehören.

Bücher

Jedes Buch besitzt einen EAN13-Barcode, der im wesentlichen seiner ISBN-Nummer entspricht. Obwohl diese Kodes nicht den Aufbau der Barcodes dieser Arbeit haben, würde ein Einbezug der EAN13-Barcodes einen Mehrwert schaffen.

In Vorlesungen oder Übungen empfehlen Professoren oder Assistenten oft bestimmte Bücher als weiterführende Literatur. Meistens nehmen sie ein Exemplar mit, in welchem die Studenten während der Pause blättern können, um selbst einen Eindruck zu erhalten. Das ETHOC-System könnte EAN13-Barcodes unterstützen, indem es die Benutzer automatisch zur Seite der ETH-Bibliothek umleitet, wo sie das Buch reservieren könnten. Eine andere Umleitung wäre zu einer Buchhandlung, wo das Buch gekauft werden kann. Werden EAN13-Barcodes unterstützt, so muss überlegt werden, ob nicht der Benutzer selbst bestimmen kann, was mit dem eingelesenen Barcode geschieht, resp. welche Bibliothek, Buchhandlung oder allgemein: welche Internetseite die aus dem Barcode geschlossene ISBN-Nummer weiterbearbeiten soll.

2.2 Verallgemeinerung auf beliebige Gegenstände

Nicht nur Dokumente, sondern auch Gegenstände können virtuelle Gegenstücke aufweisen.

Zum Beispiel könnte ein Raum via virtuellem Gegenstück von einem Benutzer online reserviert werden. Das virtuelle Gegenstück eines elektronischen Gerätes könnte für Benutzer eine Bedienungsanleitung oder eine Liste der häufigsten Probleme samt Antworten anbieten. Für spezielle Benutzer (z.B. Serviceangestellte) könnte es die letzten Reparaturen und Inspektionen anzeigen. Verbrauchsprodukte könnten nachbestellt werden.

Die Frage ist, wie die Gegenstände mit der ID versehen werden können. Das Drucken des Barcodes auf eine selbstklebende Etikette wäre eine Möglichkeit Es muss aber geklärt werden, wie man diese IDs vor Manipulation schützt. Bei Dokumenten, fällt eine Manipulation an der ID optisch auf, da der Barcode auf dem Dokument gedruckt ist. Wird der Barcode auf ein Objekt nach dessen Herstellung aufgebracht, kann man diesen Barcode leicht entfernen und durch einen

anderen ersetzen. Das Problem des nachträglichen Anbringen von Barcodes fällt bei kleinen Gegenständen ins Gewicht, bei denen die Anbringfläche sehr gering ist.

Kapitel 3

Technologien und Plattformen

Dieses Kapitel befasst sich mit den heute vorhandenen Technologien zur automatischen Identifikation von Objekten mit Fokus auf Barcodes, da diese in dieser Arbeit verwendet werden. Es werden die Lesegeräte und die Kombinationsmöglichkeiten mit Klienten-Plattformen aufgezeigt.

3.1 Identifikationstechnologien

Anglin schreibt in [5], dass die Absicht der mobilen Barcode-Leser das Verknüpfen von gedrucktem Material zum Internet sei. Barcodes stellen also eine Technologie dar, um Objekte mit einem virtuellen Gegenstück zu verbinden. Sie wird in Abschnitt 3.1.1 behandelt. Alternative Möglichkeiten, wie z.B. RFID-Marken, werden im Abschnitt 3.1.2 vorgestellt. Barcodes und RFID-Marken werden in Tabelle 3.1 verglichen.

Eigenschaft	Barcode	RFID
Kosten pro Marke	gratis	0.5 bis 1 Euro
Sichtkontakt beim Erfassen	notwendig	nicht notwendig
Gleichzeitig lesbare Marken	1	leserabhängig
Schreiben von Information	nicht möglich	beschreibbare Marke teurer
Erstellung	auf Objekt drucken	programmieren und anbringen
Infrastruktur für Dokumente	Büro-Drucker und Papier	Spezialdrucker, der Marke programmiert

Tabelle 3.1: Vergleich Barcode vs. RFID-Marken

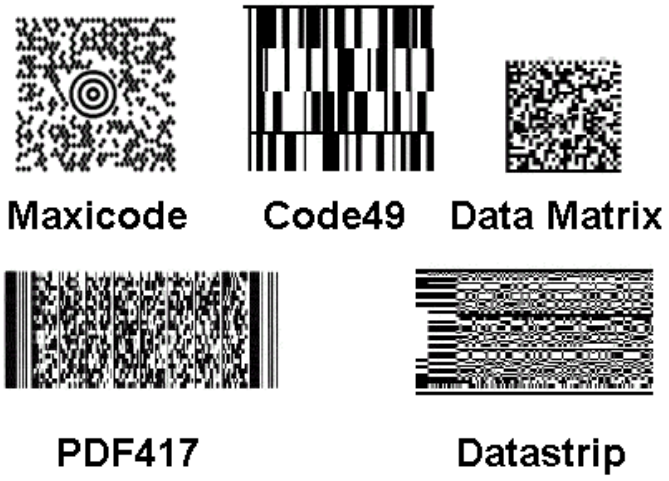

Abbildung 3.1: Beispiele zweidimensionaler Barcodes

3.1.1 Barcodes

Barcodes sind Zahlen- und Zeichenfolgen, welche in Form von Streifen aufgedruckt und somit maschinell lesbar werden. Sie werden mit speziellen Lesern erfasst, die zur Abtastung eine Lichtquelle verwenden. Dabei gibt es lineare und zweidimensionale Barcodes. Die Abbildung 3.1 (Quelle: [6]) zeigt Beispiele für zweidimensionale Barcodes. Am weitesten verbreitet sind die linearen Barcodes. Sie sind heute standardisiert auf kommerziellen Produkten zu finden.

Lineare Barcodes werden eindimensional von rechts nach links oder links nach rechts eingelesen. Zweidimensionale Barcodes sind rechteckige oder quadratischen Muster und müssen gleichzeitig in zwei Dimensionen gelesen werden. Dadurch können auf der gleichen Fläche bis zu 50 mal mehr Daten gespeichert werden als mit linearen Barcodes. Für die zweidimensionalen Barcodes braucht es teurere Lesegeräte und teilweise spezielle Druckertechnik.

Geschichte des Barcodes

Nach Russ [7] war der erste Barcode eine Entwicklung von Norman Joseph Woodland und Bernard Silver. Er besteht aus weissen Linien auf schwarzem Grund, welche einen festen Abstand zueinander haben. Information wird durch die An- oder Abwesenheit der Linien kodiert. Diese Idee floss 1952 in ein US-Patent ein.

1969 bat die NAFC (National Association of Food Chains) die Firma Logicon Inc., einen industrieweiten Standard für Barcode-Systeme zu entwickeln. Der entwickelte Standard hiess Universal Product Code (UPC). 1974 wurde der erste UPC-Scanner in Betrieb genommen. Das UPC-System ist in den USA immer noch im Einsatz.

Abbildung 3.2: UPC Version A und EAN13

Universal Product Code

UPC Version A besteht aus zwölf Ziffern (siehe Abbildung 3.2 links). Die erste Ziffer bestimmt das Nummernsystem. So steht die Zwei für gewogene Ware (wie Früchte in der Selbstbedienung), die Drei für pharmazeutische Produkte und die Null, Sechs und Sieben sind Standard-UPC-Codes. Die nächsten fünf Ziffern dienen zur Herstelleridentifizierung. Sie werden durch das Uniform Code Concil (UCC) vergeben. Für den Hersteller ist dies mit einem Jahresbeitrag verbunden. Die nächsten fünf Ziffern können durch den Hersteller selbst vergeben werden und bezeichnen das Produkt. Es ist Sache des Herstellers sicherzustellen, dass keine zwei verschiedene Produkte den selben Barcode tragen. Die letzte Ziffer ist eine Prüfsumme. Einzelne Details beschreibt Brain in [8]. Die Spezifikation ist in [7] zu finden.

European Article Numbering System

Der Wunsch nach einer Zusatzziffer im UPC zur Identifikation von Staaten wurde in der EAN13-Symbolik umgesetzt. Entgegen dem Wort “European” in seinem Namen kommt EAN13 weltweit zum Einsatz. Zum Beispiel haben ISBN-Nummern den Länderkode 978. Dadurch besitzen auch amerikanische Bücher EAN-Barcodes. (Weitere Informationen zu den Länderkodes sind unter [7] aufgeführt.)

EAN und UPC besitzen eine unterschiedliche Kodierung der Ziffern. Auch die Darstellung für das manuelle Lesen ist unterschiedlich angebracht (siehe Abbildung 3.2). EAN ist in [7] spezifiziert.

Barcodes-Kodierungsformen

Nebst den obigen Standards kann jeder eigene Barcodes beliebiger Länge generieren. Beliebig heisst, dass die Länge bis zu einem im entsprechenden Standard festgehaltenem Maximum gewählt werden kann. Es gibt Kodierungen, die das Alphabet oder den ganzen ASCII-Zeichensatz umfassen.

Der Code 128, ist laut [7] ein Code mit sehr hoher Dichte, der fähig ist, 128 ASCII-Zeichen zu kodieren, und beliebige Länge besitzen kann. Am Ende der Daten (siehe Abbildung 3.3 (Quelle: [7])) ist eine Prüfsumme vorgesehen. Es gibt drei unterschiedliche Start-Zeichen: Start-Kode A kodiert alle standardisierten alphanumerischen Tastatur-, Sonder- und Kontrollzeichen. Start-Kode

Abbildung 3.3: Aufbau eines Code 128 Barcodes

Abbildung 3.4: Beispiel eines Code 39 Barcodes

B kodiert alle standardisierten alphanumerischen Tastatur- und Sonderzeichen und unterscheidet zwischen Gross- und Kleinschreibweise. Start-Kode C kodiert nur Ziffern. Hierfür verwendet er für zwei Ziffern ein Barcode-Zeichen. Innerhalb des Datenbereichs kann durch Verwendung der Zeichen CODE und SHIFT zwischen den Kode-Arten gewechselt werden. Code 128 ist ebenfalls in [7] spezifiziert.

Code 39 ist laut [7] ein alphanumerischer Barcode, der 26 Grossbuchstaben 10 Ziffern und sieben Sonderzeichen kodieren kann. Ein Sonderzeichen ist der Stern "*" welcher den Anfang und das Ende des Barcodes markiert (siehe Abbildung 3.4). Der Stern dient lediglich diesem Zweck und kann somit nicht in den Daten auftreten. Eine Prüfsumme ist nicht verlangt, kann aber von Applikation zu Applikation selbst eingeführt werden. Code 39 ist ebenfalls in [7] spezifiziert.

3.1.2 Weitere Tagging-Systeme

Barcodes werden auf Dinge oder Dokumente aufgedruckt. Sind sie einmal aufgedruckt, repräsentieren sie die in ihnen kodierte Zeichenfolge. Diese Zeichenfolge wird von Lesegeräten ausgelesen. Barcodes sind demzufolge passiv. Es gibt aber andere Technologien, die ihre gespeicherte Information anpassen und diese selbst aktiv kommunizieren können.

In diesem Abschnitt werden die heute bestehenden Technologien aufgeführt. Auf die RFID-Tags wird spezieller eingegangen, da sie in der Literatur [2, 10] teilweise als Nachfolger der Barcodes gehandelt werden.

RFID-Tags

RFID steht für Radio Frequency Identification. Ein RFID-Tag ist ein Prozessor mit einen Speicher und einer Antenne zur Kommunikation. Es wird zwischen passiven und aktiven RFID-Tags unterschieden. Die aktiven Tags besitzen eine Batterie, die sie mit Energie speist. Sie sind mächtiger (können z.B. als portable Datenbank benutzt werden), aber auch teurer und grösser. Die passiven RFID-Tags beziehen ihren Strom durch das elektromagnetische Feld, welches ein Lesegerät erzeugt, das mit ihnen kommuniziert. Sie sind heutzutage so flach, dass sie z.B. in Papier einlaminiert werden können. Gewisse RFID-Tags können nicht nur ausgelesen, sondern auch beschrieben werden. Dies ist ein Grund weshalb Boonsor [10] glaubt, dass die RFID-Tags die Barcodes ablösen werden. Ein wesentlicher Vorteil ist, dass keine Sichtverbindung zwischen Lesegerät und RFID-Tag bestehen muss. Darin und in den ständig sinkenden Preisen der RFID-Tags sieht Mattern in [2] das Potential zur Ablösung von Barcodes. Trotz den Vorteilen, die RFIDs bringen, haben sie auch Nachteile. Deshalb schreibt Donalies in seinem Kommentar [9], dass RFID und Barcodes keine konkurrierenden Techniken sind, sondern sich in idealer Weise ergänzen.

RFID-Tags gibt es in unterschiedlichster Form, als selbstklebende Etikette oder eingepackt in einer Glashülle, welche das Anbringen an metallische Gegenstände oder das Implantieren unter der Haut von Tieren ermöglicht. (Siehe auch Langheinrich et al. [3] auf Seite drei und zehn). Die selbstklebenden Etiketten sind heute zu einem Preis von 0.5 bis einem Euro pro Stück erhältlich – mit fallender Tendenz.

Infrarot-Beacons

Im CoolTown-Projekt [11] sorgen Infrarot-Beacons für die Vermittlung der Webpräsenz eines physischen Objektes. Die Beacons werden in der Nähe des Objektes platziert und verbreiten in einem Intervall von wenigen Sekunden via Broadcast und Infrarottechnologie die in ihnen gespeicherte Zeichenkette. Die Wahl Infrarot-Technologie zu benutzen, wird durch die eingebauten Infrarot-Kommunikationsmöglichkeiten vieler Kleinstrechner, wie z.B. den PDAs, gerechtfertigt.

Ultraschall

Die Identifizierung ist auch via Ultraschall und Ultraschall-Beacons möglich. Auf der Homepage [12] wird ein Versuch erläutert, in welchem Fische ein Ultraschall-Beacon implantiert erhielten, um ihr Verhalten bezüglich künstlichen Riffs zu untersuchen.

3.2 Barcode-Leser

Die einfache Handhabung der Erstellung von Barcodes für normale Benutzer hat zu einem breiten Einsatz dieser günstigen Technologie geführt. Aufgrund dieser Verbreitung sind verschiedene Typen von Barcode-Lesegeräten auf dem Markt.

Ein Typ sind die Leser, welche mehrere Barcodes speichern können und diese später auf den Rechner übertragen. Diese Leser werden über ein Kabel (z.B. USB oder seriell) an den Rechner angeschlossen. Die Kodes werden über eine mitgelieferte Software ausgelesen. Es können auch eigene Applikationen geschrieben werden. Zu diesem Zweck sind die Schnittstellen zum Leser dokumentiert.

Andere Leser kommunizieren drahtlos (z.B. über Bluetooth) mit dem Rechner. Barcodes können so direkt an der Rechner geschickt werden. Gewisse Leser wiederum sind für spezielle Rechner gebaut. Sie können über Steckplätze an Laptops, Kleinstrechner oder Mobiltelefone angeschlossen werden. Es kommen mittlerweile immer mehr Kleinstrechner mit integriertem Barcode-Leser auf den Markt.

Betrachtet man die Vielfalt der Barcode-Leser und deren unterschiedliche Art die Barcodes zu übermitteln, stellen sich wichtige konzeptuelle Fragen fürs ETHOC-System. Die Integration unterschiedlicher Barcode-Leser soll möglich sein, ohne regelmässig das ETHOC-System anpassen zu müssen.

3.3 Klienten-Plattformen

Nicht nur die Barcode-Leser sondern auch die Betriebssysteme und die Rechner, die benutzt werden, können unterschiedlich sein. Für einige Plattformen sind wiederum nur bestimmte Barcode-Leser möglich. Dieser Abschnitt zeigt Kombinationsmöglichkeiten, die mit den Scannern möglich sind. Nicht alle Operationen, die ein virtuelles Gegenstück eines Dokumentes anbietet, können auf jedem Gerät und Betriebssystem ausgeführt werden. Es muss ein Weg gefunden werden, dass einem Benutzer nur jene Operationen angezeigt werden, die auf seinem Rechner funktionieren.

3.3.1 PC

Da der PC grössenbedingt stationär ist, kombiniert man ihn am besten mit einem Barcode-Leser, der Barcodes sammelt und später zum PC übermittelt. Der PC muss einen Internetanschluss besitzen. Das virtuelle Gegenstück kann aufgrund der Rechenleistung des PCs alle Aktionen durchführen. Lediglich fehlende Hardware könnten Aktionen verhindert. Zum Beispiel kann man nicht mit jedem PC telefonieren.

3.3.2 Laptop

Die optimale Kombination wäre das Zusammenspiel des Laptops mit Zugang zum Wireless-LAN und einem drahtlos kommunizierenden Scanner. Wie beim PC sind hier ebenfalls aufgrund der Rechenleistung (fast) alle Aktionen durchführbar.

3.3.3 PDA

Für den PDA gibt es zwei mögliche Einsatzgebiete. Ist nicht schon bereits ein Barcode-Leser eingebaut, kann entweder ein drahtloser verwendet oder einer

aufs Gerät aufgesteckt werden. Der PDA muss Zugang zum Internet haben. Es gilt abzuklären, welche Aktionen auf einem PDA funktionieren können und ob dies für alle PDAs identisch oder abhängig vom PDA-Hersteller ist. Ist letzteres der Falls, muss abgeklärt werden, wie die Geräte von ETHOC-System erkannt werden können, damit sie nur diese Aktionen angeboten bekommen, die sie auch ausführen können.

3.3.4 Mobiltelefon

Für die neuesten Ericsson-Modelle gibt es seit Winter 2001 einen Steckaufsatz der Firma AirClic. Da die Rechenleistung eines Mobiltelefons stark begrenzt ist, stellt sich hier die Frage, ob überhaupt auf den Mobiltelefonen Software-Komponenten des ETHOC-Systems laufen können. Eventuell muss hier alles online über WML-Seiten abgewickelt werden. (Bei AirClic wird dies so verlangt.) Neuerdings verstehen einige Mobiltelefone auch HTML. Es muss abgeklärt werden, ob das ETHOC-System die Mobiltelefone unterscheiden können muss oder generell WML-Seiten generieren soll.

Kapitel 4

Systemanalyse und -Anforderungen

Dieses Kapitel beginnt mit Szenarien für die Benutzer des Systems. Dadurch werden die Kernanforderungen an das System offensichtlich. Die Anforderungen ziehen benötigte Systemkomponenten nach sich. Somit wird das System als Ganzes und auf Stufe seiner Komponenten analysiert. Es sollen hier weitere Problemfelder erläutert und in den folgenden Kapiteln diskutiert und umgesetzt werden.

4.1 Benutzungsszenarien

Dieser Abschnitt illustriert Musterbeispiele für Handlungsabläufe für die Benutzer des ETHOC-Systems. Diese Beispiele zeigen die Benutzerbedürfnisse ans System und können später zur Verifikation des Systems herangezogen werden.

4.1.1 Filmveranstaltung und drahtlose Kommunikation

Dieses Benutzungskonzept sieht vor, dass der Benutzer über einen Laptop mit Anschluss ans Internet verfügt. Im Rahmen des Projektes "Neptun" werden Studierende an der ETH mit persönlichen Laptops mit Zugang zum Wireless-LAN der ETH ausgerüstet. Mit Hilfe eines Barcode-Lesers der drahtlos mit dem Notebook kommuniziert, sieht ein mögliches Szenario wie folgt aus:

Ein Student sieht auf einem Plakat, dass demnächst ein Film vorgeführt wird, den er im Kino leider verpasst hat. Er möchte diese Chance nutzen und unbedingt hingehen. Er liesst den Barcode mit seinem Leser ein, welcher sofort zum Computer übertragen wird. Es wird die Applikation gestartet, welche die virtuellen Gegenstücke anzeigt. Als vorsichtiger Mensch hat er als Standardeinstellung alle automatischen Aktionen unterbunden. Somit sieht er in einem Fenster der Applikation, weitere Angaben zum Plakat selber, sowie welche Aktionen mit diesem Plakat verbunden sind. Dort sieht er die Funktion, welche ihm erlaubt, den Termin in seine Agenda einzutragen. Er klickt auf diese Funktion

und der Filmbesuch ist eingetragen. Da er den Film kennt, interessieren ihn die anderen Funktionen, wie Zusammenfassung des Inhalts, Trailer und Link zur Filmhomepage nicht. Er löscht nun den Barcode aus der Applikation und schliesst diese.

4.1.2 Vorlesung und Studenten-Raum

Da nicht alle Studenten über einen Laptop verfügen, konzentriert sich dieses Szenario auf den PC in den Übungsräumen der ETH resp. dem eigenen Rechner zuhause und einen Barcode-Leser, der Barcodes zwischenspeichern kann.

Ein Student, der zu spät zur Vorlesung erschienen ist, hat kein Skript mehr erwischt. Um es nachträglich selber ausdrucken zu können, liest er mit seinen Barcode-Leser den Barcode vom Skript seines Bank-Nachbarn ein. Auf dem Übungsblatt, das gleichzeitig verteilt wurde, sieht er, dass man ein Kode-Gerüst als Ausgangslage benutzen kann. Um sich nicht die URL merken zu müssen, liest er auch hier den Barcode ein.

Nach dem Mittagessen geht er in einen Studentenraum und schliesst seinen Barcode-Leser an den neben dem PC zur Verfügung stehenden Steckplatz an. Es wird automatisch die Applikation gestartet, welche die virtuellen Gegenstücke anzeigt. Die Barcodes werden zu ihr transferiert und vom Barcode-Leser gelöscht. Er sieht im Fenster der Applikation untereinander alle Barcodes, die er seit dem letzen Benutzen der Applikation gesammelt hat (in diesem Fall zwei). Die Barcodes werden durch weitere Angaben ergänzt resp. ersetzt. Er sieht für jedes Objekt alle Aktionen, die möglich sind. Da er standardmässig angegeben hat, alle Aktionen, die Downloads beinhalten, automatisch ausführen zu lassen, werden das Skript, die Aufgabenstellung und das Kode-Gerüst zur Aufgabe ohne sein Zutun in ein bei der Konfiguration angegebenes Zielverzeichnis kopiert. Beim Barcode des Vorlesungsskriptes sieht er noch einen Link zur Vorlesungshomepage. Beim Barcode der Übung sieht er, dass es noch eine Musterlösung gibt. Dieser Download ist aber als noch nicht freigegeben markiert. Da er diese später downloaden will, löscht er diesen Barcode nicht resp. speichert ihn.

Wenn er das nächste Mal die Applikation startet, sieht er alle gespeicherten Barcodes wieder. Beim Programmstart wird nachgeprüft, ob und wenn ja welche Updates für die alten Barcodes bereitstehen. Gibt es neue Aktionen, so wird der Benutzer darauf hingewiesen. Bei der Übung hat das System festgestellt, dass eine neue Version der Aufgabenbeschreibung online ist. Da war also doch eine falsche Beschreibung abgedruckt. Die neue Version wird automatisch heruntergeladen, da ja die Applikation von Studenten so konfiguriert wurde, dass alle Downloads ausgeführt werden. Die Musterlösung ist natürlich noch nicht online, doch sobald dies der Fall sein wird, wird sie automatisch heruntergeladen werden.

4.1.3 Aktionen auf Kleinstrechner

Dieses Szenario sieht vor, dass der Benutzer einen Kleinstrechner (z.B. PDA) hat, welcher mangels Rechenleistung nicht alle Aktionen durchführen kann. (Z.B. gibt es nicht für alle PDAs Software, um PDF-Dokumente anzuzeigen.)

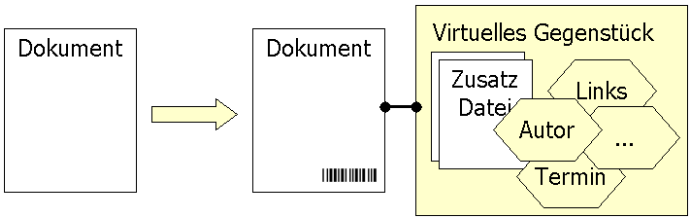

Abbildung 4.1: Prinzip des Erstellungsprozesses

Viele vorhandene Produkte benötigen heute viel Rechenleistung, was sie nicht auf jedem Gerät einsetzbar macht. Ein Student möchte einen Talk besuchen, an dem sein Lieblingsprofessor referiert. Er liest mit seinem PDA den Barcode und es wird die Applikation gestartet, welche die Barcodes verwaltet. Der Student sieht auch hier alle Aktionen, die zum Barcode gehören. Einige Aktionen sind nicht aktivierbar. Er kann sie nicht ausführen, da sein Gerät diese nicht unterstützt. Zum Beispiel kann sein PDA keine PDF-Dokumente anzeigen, jedoch die Veranstaltung in seine Agenda einfügen.

4.1.4 Annonce und Mobiltelefon

In diesem Szenario hat ein Student eine Wohnungsanzeige gesehen. Das Einlesen des Barcodes zeigt ihm eine WML-Seite an. Alle möglichen Aktionen sind Links. Da ein Mobiltelefon keine Dokumente runterladen und auch keine PDF-Dateien anzeigen kann, kann der Student leider nicht die Grundrisse sehen. Er sieht aber, dass es diese Dateien gibt und dass der sie z.B. mit dem PC betrachten kann. Das sein Mobiltelefon das WTA-Interface (Wireless Telephony Application) unterstützt, kann er die Telefonnummer der Vermieterin, welche als Kontaktperson angegeben ist, anklicken und sein Mobiltelefon wählt die Nummer automatisch. Des weiteren kann er den Orientierungs-Termin, der für alle Interessenten gilt, in seine Agenda eintragen.

4.2 Autorenunterstützung

Dieser Abschnitt beschäftigt sich mit den Autoren von Dokumenten und den Diensten der Infrastruktur, die diesen zur Verfügung gestellt werden muss.

4.2.1 Erstellungsprozess

Im Erstellungsprozess (siehe Abbildung 4.1) möchte der Autor sein Dokument, das er in einer Software seiner Wahl erstellt hat, um ein virtuelles Gegenstück aufwerten, welches auf sein Dokument zugeschnittene Zusatzinformationen oder Aktionen anbietet. Den Umfang des virtuellen Gegenstücks (Zusatzdateien, Termine für Agenden, Links etc.) kann er selbst bestimmen. Welche Aktion für welche Dokumente sinnvoll ist, ist in Abschnitt 2.1 untersucht worden. Am Ende des Prozesses ist sein physisches Dokument um einen Barcode erweitert. Dieser

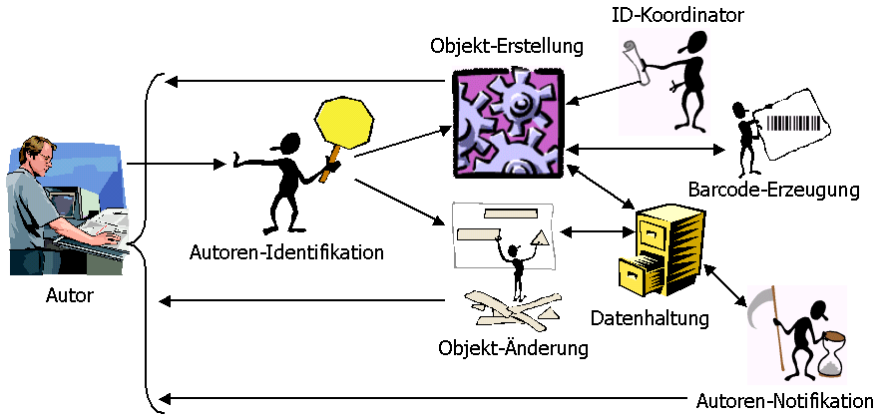

Abbildung 4.2: Zusammenspiel der autorbezogenen Systemkomponenten

dient als Verbindung zum virtuellen Gegenstück. Dazu gilt es diesen Prozess der Erstellung eines virtuellen Dokumentes für den Autor so einfach wie möglich zu gestalten.

4.2.2 Änderungsprozess

Ein weiteres Werkzeug für den Autor ist die Möglichkeit, das virtuelle Gegenstück an neue Bedürfnisse anpassen zu können. Doch auch das physische Dokument kann sich ändern. Dabei kann es erwünscht sein, dass trotz der Änderungen am physischen Dokument, das virtuelle Gegenstück nicht verloren geht. Sprich, das geänderte physische Dokument soll die gleiche ID haben wie das ursprüngliche. Dies ist z.B. dann wünschenswert, wenn Rechtschreibfehler bemerkt wurden. Im diesem Prozess wird der Autor bei den Änderungen des virtuellen Gegenstückes oder bei einer Verknüpfung eines neuen physischen Dokuments mit einem vorhandenen virtuellen Gegenstück, das auf eine ältere Version zeigt, unterstützt.

4.2.3 Autorbezogene Systemkomponenten

Die obigen Prozesse haben einige Aufgabenfelder des Systems angeschnitten. In diesem Abschnitt werden die Aufgaben mit Systemkomponenten in Verbindung gebracht, die diese Aufgabe lösen. Das Zusammenspiel dieser Komponenten ist der Abbildung 4.2 zu entnehmen. Welche Varianten es pro Systemkomponente gibt, wird in Kapitel 5 behandelt.

Autoren-Identifikation

Die Systemkomponente Autoren-Identifikation kann als Türsteher betrachtet werden, welcher den Zutritt zur Autoren-Software regelt. Diese wird gegenüber

den Autoren als Autoren-Portal bezeichnet. Hier geschieht eine allfällige Authentisierung und Autorisierung. Zudem dient diese Komponente als Eintrittspunkt zu den Autoren-Werkzeuge. Bei der Konzeption muss geklärt werden, ob der Autorenkreis eingeschränkt werden soll oder ob es jedem freisteht, Dokumente zu publizieren und anzupassen.

Objekt-Erstellung

Diese Systemkomponente ermöglicht dem Autor, über Eingabemasken das virtuelle Gegenstück zu erzeugen. Das virtuelle Gegenstück besteht aus Meta-Inforationen, d.h. aus Hintergrunds- und weiterführende Informationen, welche beim Benutzer als Text, Dateien, Links oder Aktionen ihre Form annehmen, die elektronisch verwaltet werden.

Hier geschieht also die Eingabe der Meta-Informationen. Denkbar wäre auch eine automatische Extraktion vom Meta-Informationen aus dem Dokument. Dies ist jedoch nicht Gegenstand dieser Arbeit. Die Meta-Informationen müssen durch den Autor eingegeben werden. Die Objekt-Erzeugung kontaktiert im Hintergrund alle benötigten Systemkomponenten, um die Verwaltung der Daten sicherzustellen. Der (Funktions-)Umfang eines minimalen Gegenstückes muss in dieser Komponente bekannt sein.

ID-Koordinator

Der ID-Koordinator stellt sicher, dass jede ID nur einmal vergeben wird und dass alle vergebenen IDs verwaltet werden, so dass dem Benutzer die dazugehörenden virtuellen Gegenstücke angezeigt werden können. Dabei muss die Konsistenz auch bei einem Absturz dieser Komponente und einem späteren Neustart garantiert werden. Die Eindeutigkeit der IDs muss erhalten werden, wenn diese Komponente zwecks Leistungssteigerung repliziert wird.

Barcode-Erzeugung

Diese Komponente erzeugt aus der ID einen Barcode. Es muss abgeklärt werden, welche Symbolik verwendet werden soll und ob unterhalb des Barcodes die ID in eine für den Menschen lesbaren Form angebracht sein soll. Ebenfalls gilt es bei der Konzeption die Frage zu beantworten, wie der Barcode auf ein physisches Dokument kommt und wie flexibel dessen Positionierung ist.

Datenhaltung

Diese Komponente verwaltet die Meta-Informationen und Online-Ressourcen und garantiert die Datenhaltung auch bei einem vollständigen Systemabsturz. Sie sorgt dafür, dass jede vergebene ID mit einem virtuellen Gegenstück verknüpft ist, oder als ungültig resp. gelöscht markiert ist. Mit zunehmender Lebensdauer des Systems stellt sich die Frage, wie lange die zu den IDs gehörende virtuelle Gegenstücke verwaltet werden sollen. Es wird deshalb ein Ablaufdatum für IDs geben, welches der Autor bestimmen und gegebenenfalls verlängern

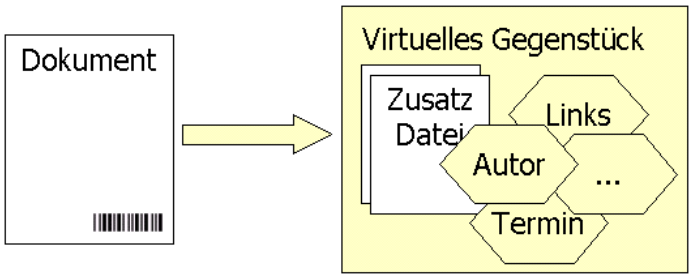

Abbildung 4.3: Prinzip des Vermittlungsprozesses

kann. Diese Komponente ist somit auch für das Löschen nicht mehr gebrauchter Daten verantwortlich.

Autoren-Notifikation

Dies Komponente informiert den Autor über Ereignisse, die das virtuelle Gegenstück seines Dokumentes betreffen. Droht die Gültigkeitsdauer von IDs bald abzulaufen, so ist es Aufgabe dieser Komponenten, den Autor z.B. via Email darauf hinzuweisen. Ein weiteres Tätigkeitsfeld dieser Komponente könnte das Führen einer Log-Datei sein, welche die Anzahl Kontaktierungen verwaltet und so dem Autor die Erstellung von Zugriffsstatistiken ermöglicht.

Objekt-Änderung

Diese Komponente ermöglicht Änderungen an den Meta-Informationen eines virtuellen Gegenstückes. Via Eingabe-Masken unterstützt es den Autor im Änderungsprozess und kontaktiert im Hintergrund die benötigten Systemkomponenten. Eine neue physische Version eines Dokumentes erhält dabei die selbe ID wie die Vorgängerversion Diese Komponente übernimmt auch die Verwaltung der Historie von elektronischen Kopien des physischen Dokumentes, sofern es vom Autor gewünscht wird.

4.3 Benutzerunterstützung

4.3.1 Vermittlungsprozess

Das Prinzip des Vermittlungsprozesses (siehe Abbildung auch 4.3) besteht darin, anhand des gedruckten Dokumentes das virtuelle Gegenstück mit seinen Komponenten und Aktionen zu vermitteln. Ein Benutzer liest hierfür den Barcode mit seinem Barcode-Leser ein. Auf seinem Rechner (PC, Laptop, PDA, Mobiltelefon etc) sieht er das virtuelle Gegenstück und seine Komponenten.

Wie dieser Prozess im Detail funktioniert hängt von den beteiligten Barcode-Lesern und Rechnern ab. Die auf dem Markt erhältlichen Barcode-Leser bieten unterschiedliche Funktionalitäten an (siehe Abschnitt 3.2). Kombiniert mit den

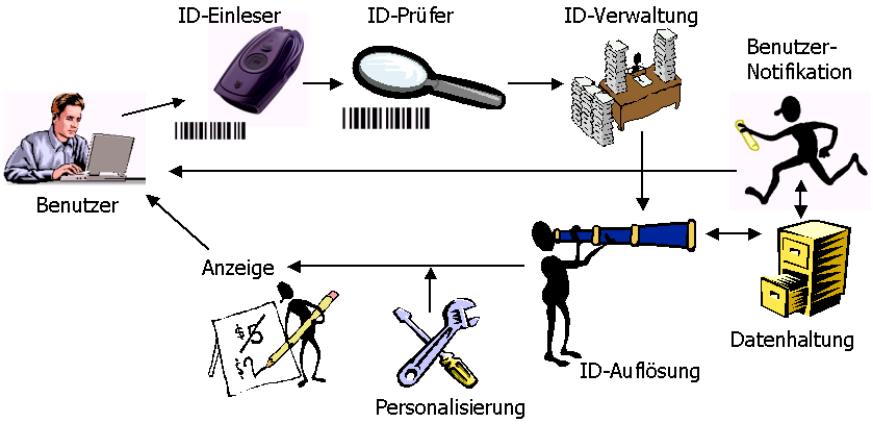

Abbildung 4.4: Zusammenspiel der benutzerbezogenen Systemkomponenten

unterschiedlichen Klienten-Plattformen (siehe Abschnitt 3.3) ergeben sich unterschiedliche Szenarien für diesen Prozess (siehe Abschnitt 4.1).

4.3.2 Benutzerbezogene Systemkomponenten

ID-Einleser

Diese Komponente liest alle IDs vom Barcode-Leser des Benutzers ein, die dieser gespeichert hat. (Kann der Barcode-Leser keine IDs speichern, so wird die aktuell gelesene ID übertragen.) Diese Komponente ist abhängig von Barcode-Leser-Typ und von der Plattform des Klienten-Rechners. Das Realisierungskonzept im Abschnitt 5.4 wird auf diese Schwierigkeit eingehen.

ID-Prüfer

Die weite Verbreitung der Barcodes ist auch deshalb möglich gewesen, weil jeder für eigene Anwendungen, ohne Kenntnisse der Standards, Barcodes erstellen kann. Ein ETHOC-Benutzer kann relativ leicht an einen fremden Barcode kommen und somit dem ETHOC-System unbekannte Barcodes einlesen.

Der ID-Prüfer hat festzustellen, ob eine ID zum ETHOC-System gehören kann. Indem er die Struktur der ID mit den Eigenschaften einer ETHOC-ID vergleicht. Es könnte aber auch nötig sein, dass bei den fremden IDs, die standardisierten, wie UPC und EAN13, erkannt werden müssen, um z.B. auf spezielle Internetseiten zu linken. Sollen fremde IDs unterstützt werden (z.B. EAN13-Barocdes, wie sie auf Bücher zu finden sind), so ist es Aufgabe des Prüfers, eine sinnvolle Umleitung auf Internetseiten zu organisieren.

ID-Auflösung

Die Datenverknüpfungskomponente hat dafür gesorgt, dass es zu jeder ID ein virtuelles Objekt oder eine Markierung, dass die ID ungültig ist, existiert. Der

ID-Prüfer hat sichergestellt, dass keine ID einen falschen Aufbau hat. Aufgabe dieser Komponente ist es das virtuelle Objekt zu vermitteln oder gegebenenfalls eine Fehlermeldung anzuzeigen.

Personalisierung

Die Präferenzen des Benutzers sollen ebenfalls verwaltet werden. So kann ein Benutzer Aktionen automatisch ablaufen lassen. Die Präferenzen zu speichern, ist Aufgabe der Personalisierung-Komponente.

Anzeige und Aktionenstart

Diese Komponente stellt sicher, dass das virtuelle Gegenstück der Klienten-Plattform entsprechend angezeigt wird. Wie sichergestellt werden kann, dass nur die Aktionen angezeigt werden, die auch auf der Klienten-Plattform ausführbar sind, muss bei der Konzeption behandelt werden. Des weiteren sollen Aktionen, die der Benutzer als automatisch ablaufend deklariert hat, ohne weiteres Zutun gestartet werden. Ob und wie dies gehen kann, soll ebenfalls bei der Konzeption geklärt werden. Ebenfalls gilt es zu klären, dass der Benutzer sicher sein kann, dass keine verborgenen Aktionen ablaufen können

ID-Verwaltung

In gewissen Szenarien kann es wünschenswert sein, eine ID für den späteren Gerauch aufzubewahren. Zum Beispiel, weil weitere Informationen für einen späteren Zeitpunkt angekündigt worden sind (etwa eine Musterlösung zu einer Übung). Eine gespeicherte ID soll auf Wunsch des Benutzer sofort nachprüfen, ob das virtuelle Gegenstück sich verändert hat. Andernfalls stellt die ID-Verwaltung beim Start automatisch fest, welche IDs neuere Inhalte aufweisen als die lokal gespeicherten. Bei einem solchen Update soll nach Möglichkeit die exakte Veränderungen angezeigt werden. Benutzer welche sich entschieden haben, eine ID verwalteten zu lassen, werden also von Änderungen am virtuellen Gegenstück in Kenntnis gesetzt, alle anderen nicht.

Eine Frage die es zu beantworten gilt, ist, ob die Verwaltung lokal oder global sein soll. In der globalen Variante verfügt der Benutzer von jedem Gerät aus die selben Barcodes. In der lokalen hat er pro Gerät eine eigene Verwaltung.

Benutzer-Notifikation

Auch auf Benutzerseite gibt es eine Notifikation, in welcher sich ein Benutzer für ein virtuelles Gegenstück einschreiben kann, um sich über Änderungen z.B. via SMS oder Email informieren zu lassen.

4.4 Zusammenfassung

Bis anhin wurden die wesentlichen Anforderungen fürs ETHOC-System zusammengestellt. Es wurden auch Fragen aufgeworfen, die zum Teil bereits beantwor-

tet werden konnten oder in den nächsten Kapiteln noch zu beantworten sind. Dieser Abschnitt fasst die Fragen und bissherige Erkenntnisse zusammen.

1. Es muss noch geklärt werden, ob es eine Registrierungspflicht für Autoren geben soll oder ob jeder direkt publizieren kann. Eine Eingrenzung der Anzahl ETHOC-Autoren auf bestimmte Endungen in der Email-Adresse ist ebenfalls denkbar.

2. Jede vergebene ID verweist auf ein virtuelles Gegenstück oder ist als ungültig markiert. Die Eindeutigkeit der ID eines Barcodes wird selbst bei einem Absturz der Komponente ID-Koordinator resp. des Gesamtsystems garantiert.

3. Die Online-Kopien müssen nicht identisch zu den physischen Dokumenten sein. Im Falle der Bestandenen-Liste im Abschnitt 2.1.2 sind sogar Online-Kopien unerwünscht.

4. Der Autor hat die Wahl, ob bei einer Änderung des physischen Dokumentes die gleiche oder eine neue ID vergeben werden soll. Allfällige verschiedene Versionen der physischen Dokumente können als elektronische Kopien in einer Historie verwaltet werden. Ergeben sich Änderungen am virtuellen Gegenstück, so fallen diese nur jenen Benutzern auf, welche die ID noch nicht gelöscht oder die sich bei der Benutzer-Notifikation eingetragen haben.

5. Die virtuellen Gegenstücke verfügen über ein Ablaufdatum, um der Speicherlast des ETHOC-Systems entgegenzuwirken. Die Autoren können bestimmen, wie viele Tage vor dem Verfallsdatum sie via Email kontaktiert werden sollen, um zum Beispiel die Gültigkeit zu verändern. Abgelaufenen Barcodes werden von der Komponente ID-Auflösung erkannt. Die Behandlung fremder Barcodes geschieht in der ID-Prüfer-Komponente.

6. Das Aussehen und die Funktionalität eines minimalen virtuelles Gegenstückes muss ebenfalls geklärt werden. Aber auch Fragen zum physischen Dokument gilt es zu klären. Wie kommt der Barcode zum Dokument und wie flexibel ist die Positionierung? Soll zusätzlich zum Barcode eine für den Menschen lesbaren Form der ID angebracht werden? Nach welchem Kodierungsstandard sollen die ETHOC-Barcodes kodiert werden?

7. Das Management der unterschiedlichen Betriebssysteme der Klienten wird in der ID-Lese-Komponente behandelt. Welche Klienten unterstützt werden sollen, wird ebenfalls in der Komponente ID-Leser behandelt. Die Anzeige-Komponente sorgt dafür, dass nur diejenigen Aktionen anzeigt werden, die auch auf der Klienten-Plattform ausführbar sind. Dabei können die Benutzer des Systems unterscheiden, ob eine Aktion möglich, (noch) nicht möglich oder neu ist. Aktionen sind auf später verschiebbar, falls IDs verwaltet werden können. Mobiltelefone können Telefonnummern als Links anbieten, sofern sie das WTA-Interface unterstützen.

8. Auf Benutzerseite sind weitere Fragen offen. Die ID-Verwaltung kann global mittels eines Profils oder lokal organisiert werden. Es gilt zu klären, welche zu einer ID gehörenden Aktionen automatisiert ablaufen. Dabei

muss darauf geachtet werden, dass keine verborgenen Aktionen ablaufen können. Das Ändern von Terminen in virtuellen Gegenstücken führt dazu, dass Einträge in den Kalendern und Agenden geändert werden. Es muss abgeklärt werden, ob der vCalendar-Standard, ein Standard zum Austausch von elektronischen Terminen, ein Verschieben von Terminen erlaubt. Wenn nicht, muss eine andere benutzerfreundliche Lösung gefunden werden.

Kapitel 5

Konzepte für das ETHOC-System

In diesem Kapitel werden mehrere Varianten zur Lösung von Problemen des ETHOC-Systems erarbeitet, bewertet und diskutiert. Am Ende dieses Kapitels wird zusammengefasst und entschieden, welche Varianten ins ETHOC-System einfliessen sollen.

5.1 Barcode-Erstellung

In Abschnitt 3.1.1 sind die beiden am häufigsten verwendeten Kodierungsformen für Barcodes vorgestellt worden. In dieser Arbeit wird der Code 128 verwendet, da er aufgrund seiner hohen Dichte weniger Platz als ein 39-Code in Anspruch nimmt. Zudem sieht dieser Standard eine Prüfsumme vor. Eine noch höhere Dichte ist zu erreichen, wenn nur Ziffern kodiert werden (siehe Abbildung 5.1). Auf zweidimensionale Barcodes wird verzichtet, weil die Lesegeräte teurer sind.

Die maximale Anzahl der Nutzdatenzeichen in einem 128er Strichkodesymbol beträgt 48 Zeichen (siehe z.B. EAN Schweiz [14]). Allerdings haben Tests mit einem Symbol-CS-2000-Barcode-Leser ergeben, dass dieser 20 Zeichen problemlos verarbeitet. Ab 21 Zeichen muss der Leser sehr präzise ausgerichtet werden, und selbst dann geht das Lesen erst nach mehrmaligen Versuchen. Die maximale ID-Länge für dieses Gerät darf 20 Zeichen nicht überschreiten. Es ist aber anzunehmen, dass andere Geräte ähnliche Schwierigkeiten haben. Aus diesem

Abbildung 5.1: Code 39 vs. Code 128

Grund ist beim Bestimmen der maximalen ID-Länge in Abschnitt 5.2 noch ein Sicherheitsabstand einzurechnen.

5.2 Objekt-ID

Um ein virtuelles Gegenstück zu einem Dokument resp. Objekt zu vermitteln, müssen beide über eine gemeinsame ID verknüpft sein. Der ID-Koordinator des ETHOC-Systems muss garantieren, dass keine zwei physischen Objekte die gleiche ID haben können. Trotz allfälligen Abstürzen oder Duplizierungen zur Leistungssteigerung der für die ID-Generierung verantwortlichen Komponenten muss die Eindeutigkeit gewährleistet bleiben.

5.2.1 Hashwert als ID

Die erste Variante spielt mit dem Gedanken, dass jeder Autor ohne Verbindung zum Internet (z.B. über eine lokale Software) in der Lage ist, einen Barcode zu beziehen. Hierzu wird ein Hashwert über das ganze Dokument gebildet, dem er als ID dient. Wenn der Funktionswert-Raum genügend gross ist, wird es keine Kollisionen geben, d.h. ist der Hashwert genügend gross, wird die Wahrscheinlichkeit, dass zwei Dokumente dieselbe ID haben, äusserst gering sein. Dennoch ist diese Möglichkeit vorhanden. Eine Änderung am physischen Dokument würde zudem zwangsläufig zu einer neuen ID und somit zum Verlust des virtuellen Gegenstückes führen. Deshalb wird diese Variante nicht in Betracht gezogen.

5.2.2 Globaler Zähler als ID

Die Idee hinter dieser Variante ist, dass global eine laufende Nummer als ID vergeben wird. Bei einem Absturz des ID-Koordinators ist sichergestellt, dass keine ID doppelt vergeben wird. Es muss lediglich nach dem Neustart die Datenhaltung nach der letzten eingetragenen ID befragt werden.

Dies funktioniert gut, solange das ETHOC-System nur auf einem Server läuft. Wird später aber entschieden, aufgrund der Lastverteilung mehrere ID-Koordinatoren zu installieren, so müssen diese mit dem Augenmerk auf gegenseitigen Ausschluss neu geschrieben werden. Die ID-Vergabe müsste sogar neu konzeptioniert werden, da ein globaler Zähler für verteilte Applikationen einen zu hohen Aufwand und ein zu hohes Verkehrsaufkommen zur Folge hätte. Mit Hinblick auf eine zukünftige Verteiltheit wird diese Variante nicht weiter in Betracht gezogen.

5.2.3 Tag-URI als ID

Tim Kindberg und Sandro Hawke beschreiben in ihrem Internet-Draft [13] eine URI (Uniform Ressource Identifier) namens “tag”, der die Eindeutigkeit der Identität ohne zusätzliche Registrierungskosten über Raum und Zeit garantiert. Der erste Teil der Tag-URI beginnt mit dem Schlüsselwort “tag:” gefolgt von

der sogenannten Tag-Autorität – einer Domain oder Email-Adresse. Nach einem Komma folgt das Datum, an dem die Tag-URI bezogen worden ist. Dieser erste Teil stellt sicher, dass sich zwei Parteien nicht in die Quere kommen. Denn jeder besitzt zu einem gewissen Zeitpunkt eindeutig eine Domain oder zumindest eine Email-Adresse. Den zweiten Teil vergibt jede Tag-Autorität autonom. Sie muss selbst sicherstellen, dass ihre Dokumente eine noch nicht vergeben ID erhalten. Hierfür sind jegliche gültigen URI-Zeichen erlaubt. Beispiele solcher Tag-URIs sind:
tag:vis.ethz.ch,2001-10-23:filmplakat
tag:nkaintan@student.ethz.ch,2001-11-01:123456789

Das Problem der Tag-URIs ist ihre Länge. Barcode-Leser können nach Typ nur eine beschränkte Barcodelänge lesen (siehe Abschnitt 5.1). Im letzten Beispiel sind 28 Zeichen allein für die Tag-Autorität verbraucht. Dies ist die Standard-Länge für Studenten-Email-Adressen an der ETH. Weitere 11 Zeichen sind für das Datum bestimmt. Möchte also ein ETH-Student ein eindeutiges Dokument veröffentlichen müssen mindestens 41 (28 + 11 + 2) alphanumerische Zeichen kodiert werden.

Die Tag-URIs gehen für ETHOC zu weit. Es muss nicht jeder Autor einen eigenen ID-Koordinator haben. Es genügt wenn einige wenige ID-Koordinatoren im Web vorhanden sind. Es ist auch anzunehmen, das die URLs der ID-Koordinatoren langlebiger sein werden als die virtuellen Gegenstücke. Also wäre die Kodierung der Zeit in die Tag-URI Platzverschwendung.

5.2.4 ETHOC-Tag als ID

In diesem Abschnitt wird eine Variante vorgestellt, welche die Vorteile der vorherigen Lösungen kombiniert. Das Hauptproblem bei den Tag-URIs ist, dass zu viele Zeichen verbraucht werden, um die Tag-Autorität zu bestimmen. In ETHOC sind die ID-Koordinatoren die Tag-Autoritäten. Die Anzahl kann vom ETHOC-System kontrolliert werden. Also kann jeder ID-Koordinator eine eindeutige ID vom ETHOC-System erhalten (z.B. aufsteigend durchnummeriert). Innerhalb seines Kontextes (seiner Koordinator-ID) kann jeder ID-Koordinator selber eindeutige IDs vergeben, indem er z.B. die Anfragen aufsteigend nummeriert. Fällt ein ID-Koordinator aus, so erfragt er beim Neustart seine eigene ID und die letzte ID, die er vergeben hat. Da die Datenhaltung persistent ist, kann er sicher sein, dass keine ID doppelt vergeben worden ist. Ist der ID-Koordinator nicht registriert, so erhält er eine neue ID und kann die Anfragen bei null beginnend durchnummerieren. Das Identifizieren des ID-Koordinator kann aufgrund seiner IP-Adresse erfolgen. Deshalb muss sichergestellt werden, dass pro Rechner keine zwei ID-Koordinatoren gleichzeitig laufen.

Ein ETHOC-Tag wird festgelegt als eine Zahlenfolge bestehend aus der Koordinator-ID (z.B. vierstellig) und der Zahl, welche der ID-Koordinator zugewiesen hat (z.B. sechsstellig). Durch eine feste Stelligkeit, sind alle Barcodes gleich lang und es kann einfacher unterschieden werden, ob der Barcode zum ETHOC-System gehört. Alle Barcodes mit einer Länge gleich zehn sind potentielle ETHOC-IDs und müssen aufgelöst werden, um festzustellen, ob es wirklich ein virtuelles Objekt gibt. Hat ein ID-Koordinator sein Kontingent erschöpft, so kann er eine neue Koordinator-ID verlangen und von vorne beginnen.

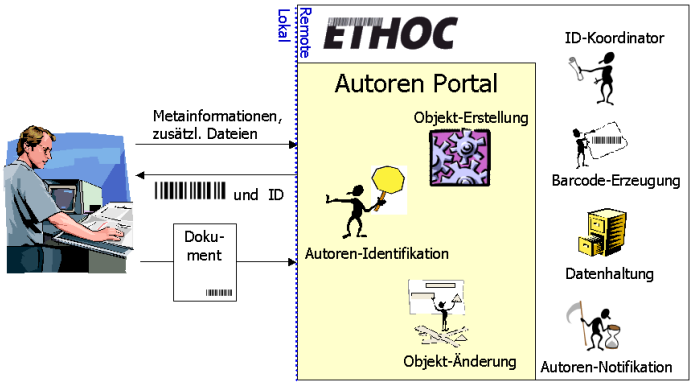

Abbildung 5.2: Variante Webservice

Die fixe Stelligkeit schafft aber ein neues Problem ähnlich der fixen IP-Adressen-Länge oder der fixen UPC-Länge; Irgendwann sind alle IDs vergeben. Eine maximale Barcode-Länge muss aber vorgegeben werden, denn die Leser können nicht beliebig lange Barcodes lesen. Im Kapitel 5.1 ist die maximale ID-Länge diskutiert worden. Eine Länge von zwanzig Ziffern erlaubt 10^{20} IDs. Dies genügt, um die Erdoberfläche 12'228 Mal mit A4-Dokumenten zu bedecken. Würde pro Sekunde eine Million Dokumente registriert, würde die Nummerierung für 3.2 Millionen Jahre reichen. In diesem Projekt ist die Lebensdauer des Systems nicht von hoher Priorität. Es soll ein Prototyp, zu Demonstrationszwecken sein. Aus diesem Grund wurde die Länge der IDs auf zehn Stellen festgelegt. Wenn pro Sekunde zehn Dokumente registriert würden, reichte die Nummerierung immer noch für 3 Jahre. Sollte das System später zum produktiven Einsatz kommen, kann eine grössere Länge festgelegt werden. Rückwärtskompatibilität wäre einfach realisierbar. Das System würde dann zwei Barcode-Längen als seine eigenen akzeptieren.

5.3 Varianten im Erstellungsprozess

In diesem Abschnitt werden drei Varianten für den Erstellungsprozess vorgestellt und die jeweiligen Konsequenzen für die System-Komponenten erläutert. Die Varianten werden evaluiert und es wird eine Variante für die Implementierung ausgewählt.

5.3.1 Webservice

In der Webservice-Variante (siehe Abbildung 5.2) ist das ETHOC-System komplett online. Nachdem der Autor sein Dokument, das er mit einem Barcode versehen möchte, fertiggestellt hat, kontaktiert er das Autoren-Portal im Internet. Er gibt seine Autoren-ID und sein Passwort ein und wird zur Objekterstellung weitergeleitet, in welcher er Angaben zu seiner Person und zum Dokument – sogenannten Meta-Informationen – machen soll. (Im Abschnitt 5.5 werden die

möglichen Metadaten und Aktionen diskutiert.) Des weiteren kann der Autor zusätzlich relevante Dokumente hochladen. Der Autor erhält einen Barcode in einem Bild-Format seiner Wahl zurück, welchen er nun in sein Dokument an eine beliebige Position zu platzieren hat, bevor er das Dokument druckt und veröffentlicht. Wünscht der Autor, dass sein Dokument auch elektronisch verwaltet werden soll, so muss er es, nachdem er es mit dem Barcode ergänzt hat, hochladen. Dazu benutzt er wieder die URL des Autoren-Portals.

Autoren-Identifikation

Die Autorenidentifikation ist für das Erstellen eines neuen virtuellen Gegenstückes nicht zwingend notwendig. Doch spätestens, wenn ein Autor ein bestehendes virtuelles Gegenstück ändern will, muss er sich als Autor ausweisen können. Durch die Authentifizierung kann dem Autor auch Arbeit abgenommen werden, in dem zum Beispiel die Autorenangaben automatisch beim Erstellen eines neuen virtuellen Gegenstückes eingefügt werden.

Die weitere Aufgabe dieser Komponente, dem Autor die vorhandenen Werkzeuge zu präsentieren und ihn an das Werkzeug seiner Wahl weiterzuleiten, kann kombiniert werden mit der Bestätigung des erfolgreichen Logins.

Objekt-Erstellung

In dieser Variante ist diese Komponente ein HTML-Formular, welches Benutzereingaben entgegennimmt. Das minimale Dokument besteht hier aus den Angaben zum Autor und zum Dokument. Eine Online-Kopie des Dokumentes kann veröffentlicht werden. Dies sollte aber erst geschehen, wenn das Dokument vom Autor mit dem Barcode versehen worden ist.

ID-Koordinator

Der ID-Koordinator wird Tags, wie in Abschnitt 5.2 besprochen, vergeben. Er ist Teil des Autoren-Portals und somit online.

Barcode-Erzeugung

Es wird der Code 128 nach Abschnitt 5.1 verwendet. Ebenfalls sinnvoll ist den Barcode in eine für den Menschen lesbaren Kodierung zu ergänzen. In dieser Variante wird der Barcode als Bild dem Autor übergeben. Dieser hat ihn an die Stelle seiner Wahl im richtigen Dokument zu platzieren. Eine Gefahr dieser Umsetzung, ist, dass der Autor die Barcodes vertauscht und in die falschen Dokumente einpflanzt. Dies sollte ihm aber nach einem Test auffallen. Der Test kann nicht automatisiert werden, da der Autor bei einem Scann-Vorgang nicht von einem normalen Benutzer unterschieden werden kann. Sobald der Autor im Besitz des Barcodes ist, kann er vom System nicht zu weiteren Tätigkeiten gezwungen werden, also auch nicht zu einem Test.

Dadurch, dass Barcodes als Bilder übergeben werden und der Autor das Bild selber ins Dokument einbaut, ist es möglich, mehrere Barcodes in ein Dokument

einzubauen. Es kann somit möglich für jeden Abschnitt in einem Dokument ein eigenes virtuelles Gegenstück erstellt werden. Es wird also weiterhin pro virtuelles Gegenstück nur einen Barcode geben. Wünscht also der Autor mehrere Barcodes für ein physisches Dokument, so muss er pro Barcode ein virtuelles Gegenstück einrichten, dass das gewünschte Verhalten modelliert.

Datenhaltung

Die Datenhaltung wird als Datenbank umgesetzt, welche auf Transaktionsbasis alle Angaben persistent speichert. Ein weiterer Teil der Datenhaltung ist das Löschen von abgelaufenen IDs und damit verknüpfter Dokumente. Dies kann in der Datenbank als zeitgesteuerter Trigger oder als eigenständiger Java-Thread implementiert werden. Im letzten Fall muss die Datenbank eine Schnittstelle zum Löschen nach aussen anbieten.

Die Angaben des Autors (zu seiner Person und zum Dokument) müssen mit der ID verknüpft werden. Diese Informationen werden strukturiert in einem XML-Dokument abgelegt, welches mit der ID verknüpft in der Datenbank abgespeichert wird. Hochgeladene Dateien werden ebenfalls im XML-Dokument vermerkt und in der Datenbank gespeichert. Das XML-Dokument bildet die Struktur des virtuellen Gegenstückes. Die genaue XML-Struktur wird im Abschnitt 7.1 erläutert. Um Angaben, die regelmässig vom System benötigt werden (wie z.B. Dokumenttitel und Ablaufdatum) schneller zur Hand zu haben, werden diese zusätzlich auch als Attribute in Tabellen der Datenbank gespeichert. Will das ETHOC-System nach abgelaufenen Dokumenten suchen, genügt nun ein Nachschauen in der Tabelle. Ohne Redundante Speicherung müssten alle XML-Strukturen nach der Informationen einzeln durchsucht werden, was sehr zeitintensiv wäre.

Autoren-Notifikation

Diese Komponente wird als Thread implementiert und prüft in regelmässigen Zeitintervallen nach virtuellen Gegenstücken, die bald ablaufen. Entdeckt sie solche, kontaktiert sie die Autoren z.B. via Email mit einem entsprechenden Hinweis.

Objekt-Änderung

Diese Komponente besteht aus Eingabefeldern für den Autor, in welchem die bereits von ihm getätigten Eingaben vorgegeben sind. Hierfür bedient sie sich des XML-Dokumentes und transformiert es anhand eines XSLT-Dokumentes in das HTML-Formular, das der Autor sieht. Änderungen werden im XML-Dokument vollzogen und die entsprechenden Dateien in die Datenbank geschrieben, resp. von der Datenbank entfernt.

Da der Autor den Barcode als Bild in sein Dokument integriert, sind Änderungen am physischen Dokument ohne Aufwand möglich, weil das Bild dabei nicht verändert werden muss. Es bleibt aber in der Verantwortung des Autors, bei substantiellen Änderungen das virtuelle Gegenstück anzupassen.

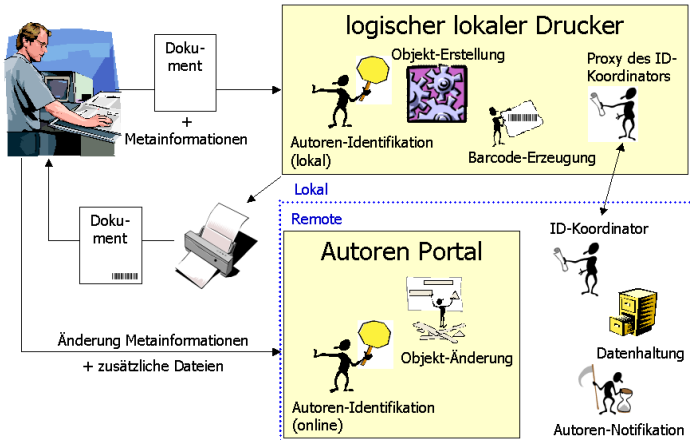

Abbildung 5.3: Variante logischer lokaler Drucker

Diese Komponente sollte, sofern es der Autor wünscht, in der Lage sein, eine Historie aller hochgeladenen elektronischen Kopien des physischen Dokumentes zu verwalten. Der Autor soll in der Lage sein, auszuwählen, welche Online-Kopien Teil der Historie sein sollen.

5.3.2 Logischer lokaler Drucker

In dieser Variante besteht das ETHOC-System aus zwei Komponenten – einem logischen lokalen Drucker, welcher eine lokale Autoren-Identifikation, die Objekt-Erstellung, die Barcode-Erzeugung und einen Proxy des ID-Koordinators umfasst und einem reduzierten Autoren-Portal, das aus online Autoren-Identifikation und Objekt-Änderung besteht(siehe Abbildung 5.3).

Ein Autor installiert den logischen lokalen Drucker als Druckertreiber auf seinem Rechner. Über den Drucken-Befehl jeder Software wird das zu druckende Dokument an den logischen lokalen Drucker weitergeleitet, der dem Benutzer eine Vorschau anzeigt und ihn um weitere Eingaben, wie freie Positionierung des Barcodes und Meta-Informationen, bittet. Ist alles erledigt, kann der Autor einen anderen Druckertreiber seines Systems auswählen, um zum Beispiel das mittlerweile um den Barcode erweiterte Dokument zu drucken oder um es in eine PDF-Datei zu transformieren. (Eine Software, die als Drucker-Treiber installiert wird, ist FinePrint[15]. Sie bietet eine Vorschau-Funktion der zu druckenden Datei(en) an. Der Benutzer kann z.B. mehrere Seiten auf eine A4-Seite verkleinern und das neue Dokument an seinen Drucker senden. Die hier vorgeschlagene Variante hätte für den Autor einen ähnlichen Benutzerkomfort.) Der logische lokale Drucker kommuniziert im Hintergrund mit dem ID-Koordinator, der weiterhin für die eindeutige Generierung der ID und Ablage der Daten in der Datenbank zuständig ist.

Bei dieser Variante muss der Benutzer während dem “Drucken” online sein. Würde man darauf verzichten und den Proxy intelligenter machen (z.B. indem er selbst IDs vergibt und die Informationen bis zu einer späteren Online-

Verbindung verwaltet), hätte man die Gefahr von Inkonsistenzen. So könnte ein Dokument ausgehängt werden, dass vom ETHOC-System noch nicht erfasst worden ist. Die Bandbreite des Internetanschlusses spielt keine Rolle, wenn davon ausgegangen wird, dass das Dokument nicht zwingend auch elektronisch verfügbar sein muss.

Will der Autor Änderungen am virtuellen Dokument durchführen, so muss er sich eines anderen Mechanismus als des Druckens annehmen. Eine Idee wäre, im Internet ein Werkzeug zum Ändern und Ergänzen von Meta-Informationen anzubieten.

Autoren-Identifikation

Für die Komponenten Objekt-Erstellung und Objekt-Änderung sind zwei unterschiedliche Autoren-Identifikationen nötig. Für ersteren lokal und für letzteren online. Somit existieren zwei Einstiegspunkte. Die Autorenidentifikation für die Erstellung eines virtuellen Dokumentes muss nicht mit einem Einloggen verbunden sein. Die Daten würden bei der Installation des logischen lokalen Druckers erhoben. Der logische lokale Drucker kann auf diese Informationen zugreifen und somit die Eingabefelder für den Autor automatisch ausfüllen. In dieser Variante ist der Autorenkreis auf diejenigen eingeschränkt, die den Drucker installiert haben.

Objekt-Erstellung

Die Erstellung der Meta-Informationen findet teilweise im logischen lokalen Drucker und teilweise online statt. Die Angaben werden im logischen lokalen Drucker erhoben. Die Koordination der beteiligten Komponenten, wie ID-Vergabe Datenverknüpfung und -Ablage geschieht online. Der Barcode kann lokal oder online erzeugt werden. Die Einbettung ins Dokument geschieht lokal.

Das minimale Dokument besteht aus den Angaben zum Autor und zum Dokument. Eine Online-Kopie des Dokumentes kann aber sofort übermittelt werden, da der Einbau des Barcodes im lokalen Drucker geschieht. Eine allfällige Übermittlung des Dokumentes inkl. Barcode setzt eine ausreichende Bandbreite voraus.

ID-Koordinator

Der ID-Koordinator wird Tags, wie in Abschnitt 5.2 diskutiert, vergeben. Er ist Teil des Autorenportals und somit online.

Barcode-Erzeugung

Die Barcode-Erzeugung geschieht wie in Abschnitt 5.1 beschrieben. Auch hier wird zusätzlich eine für den Menschen lesbaren Kodierung angebracht.

In dieser Variante wird der Barcode vom lokalen Drucker direkt ins Dokument eingebaut. Die Positionierung ist per Drag-and-Drop flexibel. Es ist aber nicht möglich, mehrere Barcodes in ein Dokument einzubauen.

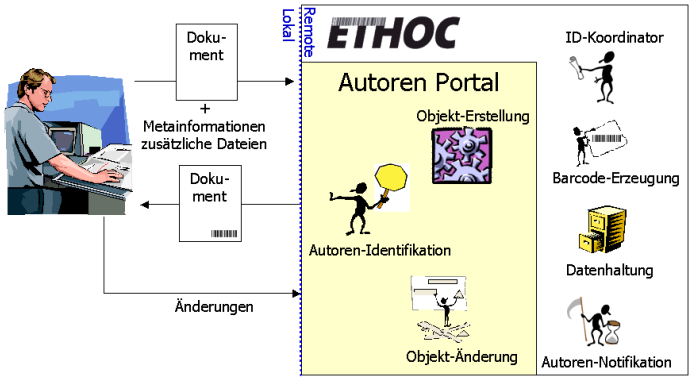

Abbildung 5.4: Variante logischer Online-Drucker

Datenhaltung

Die Datenhaltung ist, wie in der Variante zuvor (siehe Abschnitt 5.3.1), als Datenbank umgesetzt.

Autoren-Notifikation

Ist wie in der ersten Variante (siehe Abschnitt 5.3.1) zu implementieren.

Objekt-Änderung

Änderungen am virtuellen Dokument müssen über eine Internet-Schnittstelle erfolgen. Das Ändern des physischen Dokumentes erfordert eine Erweiterung des logischen lokalen Druckers. Es darf nicht immer automatisch eine neue ID vergeben werden. Es muss gefragt werden, ob der Autor eine neue ID wünscht oder ob eine an ihn vergeben ID verwendet werden soll. Hierbei muss der Autor vorsichtig sein, um nicht versehentlich den falschen Barcode in sein Dokument einzubauen.

Wie in der ersten Variante soll diese Komponente in der Lage sei, eine Historie aller hochgeladenen elektronischen Kopien des physischen Dokumentes zu verwalten. Der Autor soll in der Lage sein, auszuwählen, welche Online-Kopien Teil der Historie sein sollen.

5.3.3 Logischer Online-Drucker

Die Variante Online-Drucker sieht vor, dass der Autor zuerst eine Postscript- oder PDF-Datei erstellt. Diese kann über den Drucken-Befehl und einem geeigneten Druckertreiber geschehen. Diese Datei lädt er via eine Web-Schnittstelle zum Autoren-Portal hoch und gib die Meta-Informationen ein.

Der Barcode wird in sein Postscript-Dokument eingebaut. Der Autor kann die Position des Barcodes durch die Angabe der Ecke oder Kante (evtl. exakt unter

Eingabe von Koordinaten) beeinflussen. Er erhält das Dokument nach Auswahl entweder als PDF- oder als PS-Datei zurück. Auf Wunsch kann das Dokument gleich online hinterlegt werden.

Autoren-Identifikation

In dieser Variante wird, wie in der ersten Variante (siehe Abschnitt 5.3.1) vorgeschlagen, die Autorenidentifikation via Login implementiert.

Objekt-Erstellung

Wie in der ersten Variante (siehe Abschnitt 5.3.1) ist diese Komponente ein HTML-Formular, welches Benutzereingaben entgegennimmt.

Das minimale Dokument besteht hier ebenfalls aus den Angaben zum Autor und zum Dokument. Eine Online-Kopie des Dokumentes kann aber sofort eingebaut werden, da es sich bereits online befindet und online um den Barcode ergänzt wird.

ID-Koordinator

Der ID-Koordinator wird ETH-Tags, analog zu Abschnitt 5.2 vergeben.

Barcode-Erzeugung

Die Barcode-Erzeugung geschieht wie in Abschnitt 5.1 beschrieben. Auch hier wird zusätzlich eine für den Menschen lesbaren Kodierung angebracht.

In dieser Variante wird der Barcode online ins hochgeladene Dokument eingebaut. Die Positionierung kann grob via Angabe von Ecken, Kanten und Seiten des Dokumentes erfolgen oder evtl. flexibel via Eingabe von Koordinaten und Seitennummer. Es ist nicht möglich, mehrere Barcodes in ein Dokument einzubauen.

Datenhaltung

Die Datenhaltung ist wie in den Varianten zuvor (siehe Abschnitt 5.3.1 und 5.3.2) als Datenbank umgesetzt.

Autoren-Notifikation

Ist wie in den ersten beiden Varianten (siehe Abschnitt 5.3.1 und 5.3.2) zu implementieren.

Webservice	
⊕	Einfache Handhabung: Barcode manuell als Bild in der Anwendung des Autors positionierbar.
⊕	Physisches Dokument kann vom Autor ohne zusätzliches Werkzeug geändert werden.
⊕	Beliebig viele Barcodes in ein Dokument einbaubar.
⊙	Barcodes losgelöst von Dokumenten möglich. D.h. sie können auf Etiketten gedruckt werden und an Objekte angebracht werden.
⊖	Verwechslungsgefahr von Barcodes: Barcode wird als Abbildung durch den Autor eingebaut.
⊖	Nicht jede Anwendung erlaubt das Einbetten von Bildern (z.B. reine Text-Datei).

Tabelle 5.1: Evaluation Webservice

Logischer lokaler Drucker	
⊕	Einfache Handhabung: Autor muss nur drucken.
⊕	Einfache exakte Positionierung mittels Drag-and-Drop.
⊖	ID-Verschwendung durch unachtsame Benutzer. (Das selbe Dokument mehrmals drucken.)
⊖	Anbringen falscher Barcodes durch unachtsame Benutzer im Falle einer Änderung
⊖	Kein gemeinsamer Einstiegspunkt für die Autoren-Werkzeuge. (Der Mechanismus, um Metadaten einer bestehenden Registrierung zu ändern, kann nicht über das Ducker-Paradigma erledigt werden.)
⊙	Drag-and-Drop-Positionierung des Barcodes zu aufwendig als Bestandteil dieser Diplomarbeit.
⊙	grobe Positionierung durch Verwendung des Fine-Print-API möglich, aber dann muss eine lizenzierte Version bei jedem Klienten installiert werden (nur unter Windows möglich).

Tabelle 5.2: Evaluation lokaler Druckertreiber

Objekt-Änderung

Diese Komponente ist, wie in der ersten Variante (siehe Abschnitt 5.3.1), ein HTML-Formular, um Änderungen am virtuellen Dokument auszuführen. Die alten Online-Kopien der physischen Dokumente können auf Wunsch des Autors in einer Historie verwaltet werden. Der Autor soll in der Lage sein, auszuwählen, welche Online-Kopien Teil der Historie sein sollen.

5.3.4 Varianten-Evaluation

Die Stärken und Schwächen der einzelnen Varianten sind in den Tabellen 5.1, 5.2 und 5.3 festgehalten.

Der Vorteil des einfachen Druckerparadigmas der Variante logischer lokaler Drucker und den damit verbundenen hohen Benutzerkomfort erweist sich als

Logischer Online-Drucker	
⊕	Einheitliche Benutzerschnittstelle zum Erfassen und zum Ändern von Metadaten.
⊕	Kein "Vertauschen" von Barcodes möglich.
⊕	Maschinelle Positionierung.
⊖	Das Dokument muss vom Autor in eine PS- oder PDF-Datei transformiert werden. Entsprechender Treiber muss installiert sein.
⊖	Positionierung über Koordinatenangaben (alternativ Angabe von Ecken oder Kanten).
⊖	Hohe Bandbreite im Moment der Barcode-Vergabe nötig: Dokument wird zum Autoren-Portal und wieder zurück geschickt.

Tabelle 5.3: Evaluation Online-Drucker

Nachteil, wenn die Autorenwerkzeuge als ganzes betrachtet werden. Es ist nicht möglich, alle Werkzeuge ins Druckerparadigma einzubauen. Somit gibt es mehrere Einstiegspunkte für den Autor. Beide Werkzeuge haben zudem ein unterschiedliches Aussehen. Aus ergonomischer Sicht gesehen sind diese Punkte schwerwiegende Nachteile. Der Realisierungsaufwand für die freie Positionierung via Drag-and-Drop dürfte den Rahmen dieser Diplomarbeit sprengen, da es mit der Programmierung eines Druckertreibers und einer Applikation zur Positionierung verbunden wäre. Das Einbinden von FinePrint erlaubt nur eine grobe Positionierung. Zudem müsste bei jedem Klienten (eingeschränkt auf Windows-Rechner) eine lizenzierte Version von FinePrint installiert sein.

Das Ändern des physischen Dokuments führt ohne weiteres Zutun zum Verlust des Barcodes und des virtuellen Gegenstückes. Da dies nicht immer erwünscht ist (z.B. bei Korrektur von Rechtschreibfehlern), muss eine entsprechende Erweiterung beim logischen lokalen Drucker erstellt werden, die es erlaubt, einen bestehenden Barcode erneut zu beziehen. Allerdings sind dann Verwechslungen möglich, wenn ein Barcode z.B. anhand des Titels des Dokumentes ausgewählt wird. Aus diesen Gründen wird der logische Online-Drucker dem logischen lokalen Drucker vorgezogen. Dieser sieht für das Erstellen und das Ändern eine einheitliche Benutzerschnittstelle vor. Zudem ist ein Verwechseln der Barcodes nicht mehr so einfach möglich. Der logische lokale Drucker kann aber später als zusätzliches Werkzeug erstellt werden, der das ETHOC-Portal ergänzt.

Setzt man voraus, dass die Studenten und Mitarbeiter der ETH einheitliche Umgebungen haben und die Arbeit an der ETH erledigt wird, fallen die beiden Negativ-Punkte "PS/PDF-Transformierung" und "hohe Bandbreite" der Variante logischer Online-Drucker weg. Es gilt also noch abzuwägen, ob die Gefahr einer Barcode-Verwechslung zugunsten der beliebigen Positionierung des Barcodes in der Applikation des Autor statt durch Koordinatenangaben resp. grobe Positionierung in Kauf genommen werden soll. Da es anzunehmen ist, dass die Autoren gerne die Kontrolle über das Layout ihrer Dokumente behalten, ist die Variante Web-Service vorzuziehen. Es ist aber auch möglich den logischen Online-Drucker als Bestandteil des ETHOC-Autoren-Portals zu implementieren. Als eigenständiges Werkzeug ist dies aber nicht zu empfehlen, da das Portal mit zu ähnlichen Werkzeugen den Autor verwirren und er es als überladen empfinden könnte .

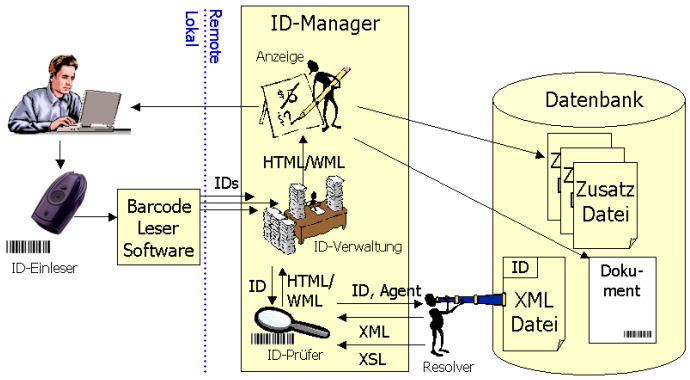

Abbildung 5.5: Variante Online-ID-Manager

In allen drei Fällen besteht das minimal mögliche virtuelle Dokument aus den Meta-Informationen. In der Variante Online-Drucker ist das Hinzufügen des Dokumentes als Online-Kopie ohne zusätzlichen Mehraufwand möglich. In keiner Variante ist es möglich, Barcodes zu beziehen, die nicht auf ein virtuelles Gegenstück verweisen.

Nach Abwägen aller Vor- und Nachteile wird die Variante Web-Service implementiert. Sei erlaubt eine freie Positionierung sowie zusätzlich Barcodes pro Absätze und sogar Barcodes losgelöst von Dokumenten.

5.4 Varianten im Vermittlungsprozess

Auch im Vermittlungsprozess sind mehrere Varianten zur Umsetzung möglich. Dieser Abschnitt vergleicht sie und stellt die Konsequenzen für die System-Komponenten dar.

Die Varianten haben gemeinsam, dass die Komponente “ID-Einleser” von der Klienten-Plattform und noch stärker vom Barcode-Leser abhängig ist, da nicht mit jedem Barcode-Leser auf die gleiche Weise kommuniziert werden kann. Der ID-Einleser muss für jeden Barcode-Leser-Typ neu geschrieben werden. Somit wird er nicht als Bestandteil des ETHOC-Systems deklariert. In dieser Arbeit wird diese Komponente exemplarisch für einige Leser geschrieben werden. Das ETHOC-System gibt eine Schnittstelle vor, die beschreibt, wie die IDs übergeben werden. Durch dieses Vorgehen ist es möglich, ohne grösseren Programmieraufwand einen Barcode-Leser ins System einzubauen. Der ETHOC-Kode bleibt davon unberührt.

5.4.1 Online ID-Manager

In dieser Variante befindet sich der ID-Manager online (siehe Abbildung 5.5). Der ID-Manager umfasst die Komponenten ID-Prüfer, ID-Verwaltung und Anzeige, setzt einen Browser beim Klienten voraus und verbraucht sonst keine

eigenen Ressourcen. Er ist somit ideal für leistungsschwache Geräte. Für Mobiltelefone ist er sogar zwingend, da auf diesen zur Zeit keine Programme installiert werden können. Mit dem Online-ID-Manager hat jedes Dokument automatisch eine eigene Homepage im Netz.

ID-Verwaltung

Neue IDs werden der ID-Verwaltung übergeben. Die ID-Verwaltung sorgt dafür, dass der Inhalt der neuen und der Inhalt von aktualisierten IDs beim Benutzer angezeigt werden kann. Die Aufgabe, zu einer ID einen passenden Inhalt zu finden, überlässt sie dem ID-Prüfer. Jede ID übergibt sie einzeln an den ID-Prüfer und erwartet einen Zeichenstrom in Form von HTML resp. WML, welchen sie zur Anzeige-Komponente weiterleitet.

Die Verwaltung der IDs ist lokal oder global möglich.
Global bedeutet: Will man Barcodes speichern, um später darauf zuzugreifen, müssten die Informationen zentral in der Datenbank abgespeichert werden. Es würde also ein Benutzerprofil erstellt. Bei jedem Auflösen einer ID müsste der Benutzer authentifiziert werden, um die Datenbank auf den neuesten Stand zu bringen oder um die Daten aus der Datenbank zu lesen. Das Abspeichern von Benutzerdaten in der Datenbank hat den Vorteil, dass Benutzer, die über mehrere Endgeräte verfügen, von allen Geräten aus auf ihre Barcodes Zugriff haben. Andererseits könnte das ständige Einloggen als Plage empfunden werden. Dies muss entschärft werden. Die Authentifizierung kann implizit ohne Eingreifen des Benutzers geschehen, indem Klienten-System-abhängige Daten übertragen werden (z.B. Seriennummer des Lesers, wie es bei AirClic der Fall ist) oder indem Benutzername und Passwort lokal gespeichert werden (z.B. Konfigurationsdatei oder Cookies). Im ersten Fall muss eine Neukonfiguration durch den Benutzer möglich sein, falls er Hardware wegen Verlust oder Defekt ersetzt. Der Nachteil der globalen Variante ist, dass der Benutzer gläsern wird; es kann ein Benutzerprofil erstellt werden.
Eine lokale Verwaltung benötigt kein zentrales Benutzerprofil und ist zum Beispiel mittels Cookies möglich. So könnten bis zu 20 IDs (so viele Cookies stehen laut Spezifikation [18] und [19] einem Server pro Klient zur Verfügung) samt Datum der letzten Änderung lokal gespeichert werden und bei jedem Kontaktieren des Online-ID-Managers neu resolviert werden, um Veränderungen am virtuellen Dokument festzustellen. Der Nachteil der lokalen Variante ist, dass nicht gesagt werden kann, welcher Bereich des virtuellen Dokumentes sich genau geändert hat. Dies wäre nur möglich, wenn man die alte XML-Struktur im Cookie abgespeichert hätte. Cookies haben aber laut Spezifikation [18, 19] eine Maximalgrösse von 4'000 Zeichen (Byte).

ID-Prüfer

Der ID-Prüfer eruiert, ob die ID dem ETHOC-Format entspricht. Auf die Überprüfung, ob es ein anderer Standardcode wäre, wird verzichtet. Alles was vom Aufbau her keine ETHOC-ID sein kann, wird als nicht auflösbar angezeigt. D.h. es wird eine Fehlermeldung in HTML oder WML erzeugt. Plausible IDs werden dem ID-Auflöser übergeben, dessen Antwort, eine XML- und eine XSL-Datei, werden transformiert und das Ergebnis an die ID-Verwaltung weitergeleitet.

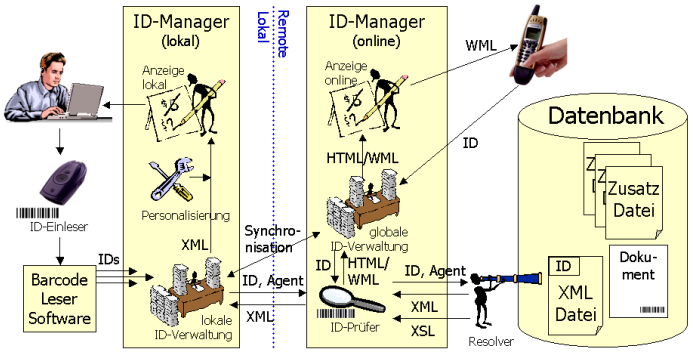

Abbildung 5.6: ID-Manager mit lokaler Komponente

Somit kann für jeden Klienten-Typen das Ausgabeformat optimiert werden. Im Falle von Mobiltelefonen wäre dies WML, in anderen Fällen z.B. HTML. Mit dieser Technologie können Aktionen, die nicht auf dem Klienten laufen können, ausgeblendet werden.

ID-Auflöser

Diese Komponente besorgt sich aufgrund der ID die XML-Struktur des virtuellen Gegenstückes aus der Datenhaltung und übergibt sie dem Prüfer. In jedem HTTP-Request wird der Browser-Typ anhand des User-Agents mitübertragen. Dies kann ausgenützt werden, um die Ausgabe den Klienten anzupassen. In dieser Variante wird aufgrund des Agententyps und der ETHOC-ID eine passende XSL-Datei aus der Datenbank besorgt werden.

Personalisierung

In dieser Variante wird alles im Browser des Benutzers dargestellt. Somit können Informationen und Aktionen entweder als Links oder als Text dargestellt werden. Die Standard-Sicherheitseinstellung der Browser verbieten automatische Downloads und den Start von Programmen. Somit ist eine Personalisierung in dieser Variante nicht möglich.

Anzeige und Aktionenstart

Die anzuzeigende Ausgabe wurde bereits durch den ID-Prüfer in der Form (HTML oder WML), die der Browser versteht, generiert. Da im Browser alle Aktionen über Links explizit aufgerufen werden müssen, können keine verborgenen Aktionen ausgeführt werden.

5.4.2 Variante ID-Manager mit lokaler Komponente

Bei dieser Variante befindet sich der Hauptteil des ID-Managers auf der lokalen Klienten-Plattform, d.h. er muss dort installiert werden. Dies würde aber

bedeuten, dass keine Mobiltelefone unterstützt werden können, da auf vielen Mobiltelefonen keine Software installiert werden kann. Aus diesem Grund besteht in dieser Variante der ID-Manager nebst der lokalen auch aus einer Online-Komponente. Der ID-Prüfer und ein Teil der Anzeige-Komponente befindet sich online. Die globale ID-Verwaltung ist ebenfalls online. Lokale ID-Verwaltung, Personalisierung und der restliche Teil der Anzeige-Komponente sind Bestandteil des lokalen ID-Managers.

Neue IDs werden der lokalen ID-Verwaltung übergeben. Diese synchronisiert sich mit der globalen ID-Verwaltung. Insbesondere werden alle neuen IDs weitergeleitet und Updates von verwalteten virtuellen Gegenstücken erkannt. Neue und zu aktualisierende IDs werden einzeln dem ID-Prüfer weitergeleitet, welcher eine XML-Antwort zurückgibt. Diese besteht aus der Struktur des virtuellen Dokumentes ohne jene Angaben, die der Autor nicht publizieren möchte oder aus einer Fehlermeldung, falls die ID nicht aufgelöst werden kann. Das Resultat wird direkt zur Anzeige-Komponente weitergeleitet.

Wird ein Klient benutzt, auf dem keine lokale Software installiert ist, so wird die ID direkt der globalenVerwaltung zugeschickt. Diese leitet die ID weiter und gibt die HTML/WML-Antwort an die online Anzeige-Komponente weiter.

ID-Verwaltung

Die übermittelten XML-Dateien können gespeichert und bei einem Update durch diese Komponente auf Veränderungen verglichen werden. Somit ist es möglich, genaue Änderungen auszumachen. Die lokale Verwaltung wird mit der zentralen Historie der globalen ID-Verwaltung abgeglichen, um alle IDs auf allen Geräten zu haben. Die globale ID-Verwaltung speichert lediglich die IDs ohne XML-Inhalt.

ID-Prüfer

Der ID-Prüfer ist identisch mit dem der ersten Variante (siehe Abschnitt 5.4.1). Hinzu kommt lediglich, dass es einen neuen Agenten-Typ, den ETHOC-Browser, gibt, der als Resultat seiner Anfrage XML erwartet.

ID-Auflöser

Diese Komponente ist ebenfalls identisch mit derjenigen aus der ersten Variante (siehe Abschnitt 5.4.1).

Anzeige und Aktionenstart

Der Online-Teil dieser Komponente ist identisch mit der ersten Variante (siehe Abschnitt 5.4.1). Im lokalen Teil wird die übermittelten XML-Struktur in eine interne Java-Struktur umgewandelt und daraus ein Layout erstellt, welches im Klienten angezeigt wird. Abhängig von der Personalisierung können Aktionen automatisch gestartet werden. Es können nur Aktionen ablaufen, die in den Autorenwerkzeugen angeboten wurden. Somit gibt es keine verborgenen Aktionen.

Personalisierung

In dieser Variante wird die Personalisierung umgesetzt. Sie ist Teil der lokalen Software und wird lokal gehalten. Mögliche Automatismen sind: Alle Dateiein runterladen, Kalenderanträge automatisch öffnen, bestimmte Dateitypen direkt öffnen, Barcodes nach beenden der lokalen Applikation sofort löschen sowie die Angabe eines Zielverzeichnises für die Ablage aller automatisch runtergeladenen Dokumenten.

5.4.3 Variantenvergleich

Beide Varianten bieten eine hohe Flexibilität bei der Klientenauswahl. Für jeden Klienten-Typ muss lediglich ein XSLT-Dokument als Konfiguration angefertigt werden. Für jedes Gerät kann direkt bestimmt werden, welche Aktionen ausführbar sind und welche nicht, indem es entweder die Darstellung lokal ausführt oder der Server es mit einer XSLT-Struktur verknüpfen kann.

Beim Online-ID-Manager ist das Layout auf XML (also HTML und WML) festgelegt. Bei der lokalen Variante ist das Layout mächtiger, dafür werden aber mehr Ressourcen beim Klienten benötigt. Die Erstellung eines Benutzerprofils bei der Datenbank des Online-ID-Managers kann sowohl als Vorteil (von jedem Gerät aus dieselben IDs) oder als Nachteil (Privatsphäre) ausgelegt werden. Auf jeden Fall ein Vorteil ist, dass die XML-Strukturen der IDs verwaltet werden. Dadurch kann bei einem allfälligen Update des virtuellen Gegenstückes, die exakte Veränderung angezeigt werden. Es müssen lediglich neue und alte XML-Struktur verglichen werden.

Ein grosser Nachteil der Online-Variante ist, dass keine Aktionen automatisch ablaufen können. Dafür muss ausser der Barcode-Leser-Software (Komponente ID-Einleser) nichts installiert werden. Ein Nachteil der Offline-Variante ist die benötigte Rechenleistung auf der Klienten-Seite. Die XML-Struktur muss bearbeitet werden, um dargestellt werden zu können.

In dieser Arbeit wird der ID-Koordinator mit lokaler und online Komponente realisiert.

5.5 Meta-Informationen

Ein elektronisches Dokument besitzt immer zusätzliche Hintergrund-Informationen (wie Titel und Autor), sogenannte Meta-Informationen, die über spezielle Befehle sichtbar gemacht werden können. Ähnlich hierzu sollen die virtuellen Gegenstücke zusätzliche Informationen verwalten. Aktionen sind Funktionen, die auf dem Rechner eines ETHOC-Benutzers ausgeführt werden. Das virtuelle Gegenstück verwaltet diese Aktionen.

Die Tabelle 5.4 listet alle im ETHOC-System verwendeten Metadaten auf. Informationen können dokumentspezifisch oder generisch sein. Zudem können sie teilweise automatisch erhoben werden. Einige Angaben sind zwingend, z.B. der Name des Autors. Die Angaben des Autors zu seiner Person können von Dokument zu Dokument variieren. Aus diesem Grund kann er entscheiden, welche Angaben von ihm für ein Dokument publiziert werden sollen und welche

Meta-Information	dok.abh.	automatisch	zwingend
Name des Autors		X	X
Organisationseinheit des Autors		X	
Emailadresse des Autors		X	
Homepage des Autors		X	
Telefonummer des Autors		X	
Faxnummer des Autors		X	
Adresse (Ort, Strasse, Nummer)		X	
Büro des Autors		X	
Was soll publiziert werden?	X	X	X
Kontaktadressen	X		
Titel des Dokumentes	X		X
Stichworte zum Dokument	X		
Dokument-Kurzbeschreibung	X		
Dokument gültig bis	X		X
Online Kopie des Dokumentes	X		
Historie des Dokumentes	X	X	
Aktionen des Dokumentes	X		

Tabelle 5.4: Meta-Informationen eines Dokumentes

nicht. Dazu gibt es für jede Meta-Information, die den Autor betrifft, ein Flag, das er setzten kann, damit die Angabe publiziert wird. Alle erhobenen Meta-Informationen werden in einer XML-Struktur verwaltet. (siehe Abschnitt 7.1).

5.5.1 Aktionen

Die in Abschnitt 2.1 beschriebenen Aktionen sind in der Tabelle 5.5 zu wenigen Aktions-Typen zusammengefasst. Zum Beispiel sind Filmkritiken, Lebensläufe, Portraits, Folien, Abstracts, Bilder und Musterlösung aus Abschnitt 2.1 normale zusätzliche Dateien, und Reservationssysteme, Anmeldeformulare, Testatübersicht, Feedbackformulare, Abstimmungen und Fragebogen aus Abschnitt 2.1 sind normale Links zu Fremdseiten oder zu Modulen des Systems.

Aktionen sind auch automatisierbar. So können automatisch Online-Kopie heruntergeladen, zusätzliche Dateien abhängig vom Typ geöffnet oder heruntergeladen oder Termine direkt mit dem Kalender-Programm geöffnet werden.

5.5.2 Module

ETHOC-Module sollen dem Autor helfen, zusätzliche Funktionalität in die virtuellen Gegenstücke seiner Dokumente zu bringen. Sie sind nichts anderes als Servlets im Internet, die aber zum ETHOC-System gehören. So könnten z.B. Veranstaltungen um Anmeldeformulare oder Vorlesungsskripte um Feedback-Formulare ergänzt werden. Auch Abstimmungen sind möglich, denn die Benutzer sind aufgrund der globalen ID-Verwaltung eingeloggt und somit dem System bekannt. Dadurch kann auch pro Benutzer-ID nur einmal abgestimmt werden.

Typ	Aktion
Online-Kopie	Herunterladen
Termin	Start (Datum und Zeit), Ende (Datum und Zeit), Adresse (Ort, Strasse und Nummer) Titel und Beschreibung in Kalender eintragen
Links	Weiterleitung zur URL
Zusätzliche Datei	Ab Zugangsdatum, für Benutzer herunterladbar oder öffnen
Benutzer-Kontakt	Email, Telefon, Adresse
Historie	Dokumente auflisten und Selektion zum Download

Tabelle 5.5: Aktionen eines Dokumentes

Dies würde aber auch eine eingehende Überprüfung des Sicherheitskonzeptes im ETHOC-System erfordern.

Durch die Modularität kann das ETHOC-System flexibel erweitert werden, ohne das Grundgerüst, wie die XML-Struktur des virtuellen Gegenstückes, ändern zu müssen. Die Modul-Anbindung ist so konzeptioniert, dass Entwickler von Modulen keinen bestehenden Kode des ETHOC-Systems anpassen müssen. Für den Benutzer sind Module normale Links ins Internet, welche ihm eine Funktionalität zum Dokument bieten. Module sollen Funktionen anbieten, die nicht einfach anderswo im Internet mit besserem Leistungsumfang gefunden und konfiguriert werden können. Diskussionsforen z.B. können bei einem anderen Anbieter ausgewählt und als Link in einem Dokument integriert werden. Im Rahmen dieser Arbeit sind exemplarisch das Feedback- und das Autorennotifikationsmodul programmiert worden, um zu zeigen, wie neue Module eingebunden werden können.

5.6 Zusammenfassung

Dieser Abschnitt fasst zusammen, welche Varianten umgesetzt werden und wie die Probleme aus den Kapiteln zuvor gelöst wurden.

1. Für die Erstellung von virtuellen Gegenstücken wird die Variante Webservice realisiert. Ein Autor muss sich zuvor im Autoren-Portal ausweisen. Dies geschieht mittels Autoren-ID und Passwort. Die Grundfunktionalitäten der virtuellen Gegenstücke werden durch Module erweitert. In dieser Arbeit wird exemplarisch das Feedbackmodul umgesetzt.

2. Als ID-Manager wird die Variante mit lokaler und Online-Komponente implementiert. Es können potentiell alle Klententypen unterstützt werden. Dabei sind die Fähigkeiten der Klientengeräte in XSLT-Dateien auf dem Server festgehalten. Den Mobiltelefonen z.B. werden WML-Seiten angeboten, hierfür wird die zugehörige XSLT-Datei aus der Datenbank ausgelesen. Neue Barcode-Lesermodelle können ohne Änderung des ETHOC-Systems eingebunden werden – es muss lediglich eine auf den Barcode-Leser zugeschnittene ID-Leser-Komponente geschrieben werden.

3. Die ETHOC-IDs haben eine feste Stelligkeit (vier Stellen für die Koordinator-ID und sechs Stellen für die Rest-ID) und werden mittels Code 128 kodiert. Fremde IDs können durch die abweichende ID-Länge erkannt werden. EAN-Barcodes für Bücher oder UPC-Barcodes für Produkte werden vorerst nicht unterstützt.

4. Es wird ein zentrales Benutzerprofil in der Datenbank geführt. Ein geändertes virtuelles Gegenstück wird im Benutzerprofil ebenfalls als geändert markiert. In der lokalen Applikation wird die XML-Struktur gespeichert, um bei einer Aktualisierung die Änderungen lokalisieren zu können. Es wurde geklärt, welche Aktionen automatisch ablaufen können und dass keine ungewollte, verborgenen Aktionen möglich sind.

5. Der vCalendar-Standard [20] erlaubt keine Verschiebungen von Terminen. D.h. hat ein Benutzer einen Termin in die Agenda eingetragen und wird dieser später durch den Autor geändert, so muss der Benutzer daran denken, den alten Eintrag zu entfernen. Vergisst er dies, so ist der Eintrag doppelt vorhanden. Es liegt also auch am Autor, dafür zu sorgen, dass bei einer Terminänderung z.B. im Begleittext des Termins die Verschiebung des Termins beschrieben ist, damit ein Benutzer mit zwei Kalendereinträgen weiss, welches der korrekte ist. Dieses Vorgehen ist äquivalent zum heutigen vorgehen. Erfährt jemand, dass sich ein Termin geändert hat, so muss er den alten Termin selbst austragen.

Zusätzlich werden im Design-Kapitel folgende Probleme gelöst:

- Die genauen Fähigkeiten der Klienten-Typen ist Teil des XSLT-Designs resp. des Designs der beiden ID-Manager.

- Das Anzeigen der genauen Änderungen eines virtuellen Gegenstückes ist Sache des lokalen ID-Managers und somit Teil seines Designs.

Kapitel 6

Design des ETHOC-Systems

Dieses Kapitel beschreibt detailliert die im Kapitel 5 gewählten Varianten und legt die Grundlagen für die Implementation des ETHOC-Systems.

6.1 XML-Strukturen und -Verarbeitung

Das ETHOC-System und die virtuellen Gegenstücke sind sehr flexibel aufgebaut und können leicht erweitert werden. Diese Flexibilität muss verwaltet werden. Hier bietet sich XML an, da strukturgestützte Informationen und der Inhalt gemeinsam abgespeichert werden. ETHOC benutzt XML für zwei Zwecke. Zur Modellierung virtueller Gegenstücke und zur Konfiguration des Systems. Um ETHOC einfach erweitern zu können, werden die Anzahl der Module und Werkzeuge nicht fest im Programm kodiert, sondern über Konfigurationsdateien in XML-Form dynamisch eingelesen. Soll zu einem späteren Zeitpunkt ETHOC um neue Werkzeuge oder Module erweitert werden, so muss kein Kode verändert werden. Lediglich die Konfigurationsdateien müssen angepasst werden. Der Aufbau der XML-Strukturen wird im Abschnitt 7.1 näher erläutert. Die Verarbeitung der XML-Strukturen tritt also in folgenden Orten im ETHOC-System auf:

- Nach erfolgreicher ID-Auflösung wird eine XSL-Transformation durchgeführt, um die Ausgabe für den Klienten zu erzeugen.
- In den Prozessen der Objekt-Erstellung und -Änderung wird aus den Eingaben des Autors die XML-Struktur erzeugt oder erweitert. Zur Anzeige der bereits getätigten Angaben wird die XML-Struktur in ein HTML-Formular transformiert.
- Beim Erstellen oder beim Ändern eines virtuellen Dokumentes wird die aktuelle Konfiguration der Module ausgelesen.
- Nach erfolgreichem Login werden die aktuell angebotenen Werkzeuge ausgelesen, um dem Autor das Weiterarbeiten zu ermöglichen.

Abbildung 6.1: Besipiel eines ETHOC-Barcodes

- Der lokale ID-Manager (die lokale Applikation wird gegenüber dem Benutzer als ETHOC-Browser bezeichnet) prüft nach, welche ETHOC-Typen zugelassen sind. In dieser Arbeit gibt es den Typ “doc” und “err”. Die XML-Objektstruktur der virtuellen Gegenstücke wird verwendet, um eine Java-Objektstruktur zu erstellen.

Die XML-Strukturen werden also gelesen, erzeugt oder transformiert. Hierfür werden Java-Bibliotheken verwendet. Zur Transformation wird XALAN [25] verwendet. Zum Parsen eines XML-Dokuments stehen zwei APIs zur Verfügung DOM und SAX, welche durch Parser wie XERCES [26] zur Verfügung gestellt werden. Bei DOM wird das XML-Dokument in eine Baumstruktur übersetzt. Dieser Baum kann traversiert und die Informationen extrahiert und manipuliert werden. DOM empfiehlt sich nur, wenn ein Dokument manipuliert oder mehrmals geparst werden muss. Zudem wird viel Speicher verbraucht, da das ganze Dokument als Baum verwaltet wird. Beim Parsen des Dokumentes mit einem SAX-Paser werden Events wie “start-Element” oder “endElement” ausgelöst, auf welche reagiert werden kann. Für das einmalige Auslesen einer XML-Struktur empfiehlt sich somit der SAX-Parser. Zum Erzeugen und Manipulieren einer XML-Struktur lassen sich nur die DOM-Schnittstellen verwenden. Da es aber relativ aufwendig ist, einen generischen DOM-Baum zu manipulieren und mit anderen Bäumen zu vergleichen, wird in ETHOC ein anderer Weg beschritten. Zu jedem Element in der XML-Struktur des virtuellen Gegenstückes gibt es eine Java-Klasse. Diese Klasse enthält, die in der Struktur definierten Instanzen und Elemente und bietet Methoden, wie z.B. die Serialisierung der internen Struktur zu einem XML-String. Weitere Infomationen zu DOM und SAX sind [16] und [17] zu entnehmen.

6.2 Barcode

Die ID eines Dokumentes ist zehnstellig und wird als Barcode mittels Code 128 kodiert. Es wird ein Bild der Grösse 29x110 Pixel erstellt (siehe Abbildung 6.1). Unterhalb des Barcodes befindet sich, in für den Menschen lesbaren Form, die Aufschrift ETHOC gefolgt von der ID in Dezimalschreibweise. Diese Aufschrift ermöglicht dem Benutzer, den Barcode zuzuordnen und die ID von Hand einzugeben.

6.3 ETHOC-Portal

Da auf Seiten des Autors und auf Seiten des Benutzers eine Lösung mit Online-Komponente zum tragen kommt, bietet es sich an, für das ganze ETHOC-

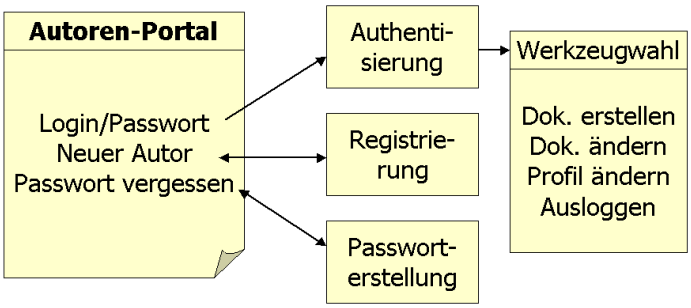

Abbildung 6.2: Einstieg ins Autorenportal

System eine Einstiegsseite in HTML zu entwickeln. Diese kann später im Prototypen z.B. unter www.ethoc.ethz.ch veröffentlicht werden. Die Grundfunktionalität, die das Portal anbietet, ist die Verzweigung zum Autoren-Portal und zur Benutzer-Online-Komponente, in welcher die ETHOC-ID im Browser von Hand eingegeben werden kann. Eine Seite, auf welcher das ETHOC-Projekt dem Benutzer vorgestellt wird, ist ebenfalls in Planung.

6.4 Autoren-Portal

Das Autoren-Portal bietet einem Autor nach erfolgreicher Anmeldung verschiedene Werkzeuge an. Innerhalb eines Werkzeuges wird der Autor in seinen Interaktionsmöglichkeiten geführt. Ein Werkzeug besteht also aus einer Reihe von HTML-Seiten, welche dynamisch aus Java-Servlets erstellt werden.

Die Java-Klassen des Autoren-Portals gehören ins Paket “Ethoc.Authoring”. Klassen, die sowohl beim Autor als auch beim Benutzer gebraucht werden, finden ihren Platz unter “Ethoc.System”. Die Klassen, die nur beim Benutzer benötigt werden, sind unter “Ethoc.Browser” untergebracht. Die Klassen des Benutzer-Portals sind im Paket “Ethoc.Resolving” zu finden. Wird das Autoren-Portal zu einem späteren Zeitpunkt um weitere Werkzeuge erweitert, so muss lediglich die XML-Datei “authoringtools.xml” angepasst werden. Die zusätzlichen Servlets müssen selbstverständlich noch hochgeladen werden. Dem Entwickler neuer Werkzeuge steht frei, neue Java-Pakete zu verwenden.

6.4.1 Einstiegsseite

Das Autorenportal beginnt mit einer HTML-Seite, in welcher der Benutzer folgende Möglichkeiten hat (siehe Abbildung 6.2):

1. Einloggen mittels Autoren-ID und Passwort (Authentisierung).
2. Link zur Erstellung einer neuen Autoren-ID (Registrierung).
3. Link zu Passworterzeugung, falls Passwort vergessen worden ist.

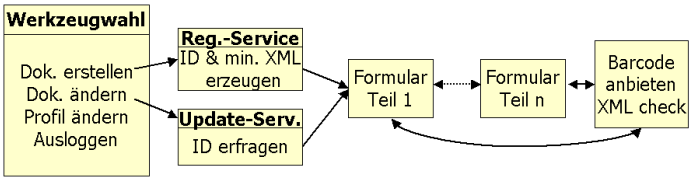

Abbildung 6.3: Ablauf der Objekt-Erstellung und -Änderung

Der erste Fall ist die häufigste Benutzungsform. Nach der Eingabe von ID und Passwort erhält der Autor, sofern seine Angaben korrekt waren, die Übersicht über alle Autoren-Werkzeuge. Von diesem Zeitpunkt an hat der Autor eine persönliche Session, die ihn über alle Seiten hinweg begleitet. In dieser Session ist seine Autoren-ID vermerkt. An die Session können später weitere Parameter, wie die ID des gerade bearbeiteten virtuellen Gegenstückes, angefügt werden. Fehlt die Session, handelt es sich entweder um einen Manipulationsversuch, eine abgelaufene Session oder um einen Browser, der keine Cookies unterstützt. Dies wird dem Benutzer mittels einer Exception in HTML-Form angezeigt.

Im zweiten Fall wird ein Formular angezeigt, in welchem sich der Autor registrieren kann und seine ID bestimmt. Das Passwort wird automatisch erzeugt und ihm an die angegebene Email-Adresse gesendet. Dadurch wird sichergestellt, dass der Autor Zugang zum Email-Konto hat, welches er bei der Anmeldung als sein eigenes ausgibt. Es kann aber nicht verhindert werden, dass die weiteren Angaben falsch sind. Sein Passwort resp. sein ganzes Profil kann er, sobald er eingeloggt ist, ändern.

Falls der Autor sein Passwort vergessen hat, kann er im dritten Fall ein neues Passwort beantragen, welches automatisch erstellt und an seine Email-Adresse geschickt wird. Hierfür muss er entweder seinen Login oder die Email-Adresse, mit der er sich registriert hat, wissen. Es kann nicht sein altes Passwort zugeschickt werden, da es in der Datenbank mit einer Einwegfunktion verschlüsselt abgelegt worden ist.

6.4.2 Objekt-Erstellung und -Änderung

Objekt-Erstellung und -Änderung haben beide eine sehr ähnliche Benutzerführung (siehe Abbildung 6.3). Hat sich der Autor nach dem Login entschieden, ein neues Dokument mit Online-Ressourcen zu verknüpfen, so reserviert das erste Servlet eine eindeutige ID und verbindet sie mit der Session. Es generiert ein minimales XML-Dokument, das die Angaben zum Autor berücksichtigt und speichert es in der Datenbank mit dem Vermerk “invalid” ab. Dies bedeutet, dass die XML-Struktur durch den Autor nicht vollständig durchgearbeitet ist und es somit noch kein virtuelles Dokument zur ID gibt. Hat sich der Autor nach dem Login entschieden, ein bestehendes virtuelles Dokument zu ändern, wird dessen Dokumenten-ID mit der Session verknüpft und das erste Servlet, welches das virtuelle Dokument bearbeitet, aufgerufen.

Um die Eingabeseite nicht zu überladen, werden die Benutzereingaben zum virtuellen Dokument über mehrere Formulare, erzeugt durch Servlets, verteilt. Da

die XML-Struktur aus vier Hauptkomponenten besteht – Angaben zum Autor, zum Dokument, zu den Kontaktpersonen und Aktionen – bietet sich eine Aufteilung der Eingabeformulare nach dieser Hauptstruktur an. (In Abbildung 6.3 als Formular Teil 1 bis n dargestellt).

Ein an der Generierung des virtuellen Gegenstückes beteiligtes Servlet zeigt einen Teil der Angaben in der XML-Struktur als Formular an. Die XML-Struktur wird aus der Datenbank gelesen und zusammen mit einem XSL-Dokument zu einer HTML-Seite transformiert. Über Schaltflächen kann der Benutzer neue Felder anfordern (z.B. einen weiteren Kontakt hinzufügen), Felder löschen oder zum nächsten Servlet navigieren. In allen Fällen werden die getätigten Angaben in der Datenbank abgespeichert.

Die Gültigkeitsdauer der Verknüpfung zwischen Dokument und virtuellem Gegenstück muss ebenfalls erfragt werden. Wird dieser Zeitpunkt überschritten, so wird die Dokumenten-ID als gelöscht markiert. Bei einer späteren Referenzierung wird dem Benutzer die erfolgte Löschung mitgeteilt. Somit lässt sich eine gelöschte ID von einer ungültigen Referenz unterscheiden. Die Konfiguration der Autoren-Notifikation kann in diesem Servlet vollzogen werden. Der Autor kann bestimmen, wie viele Tage er vor Ablauf seines Dokumentes informiert werden soll. Diese Funktionalität ist in den Formularen 1 bis n integriert worden.

Die ETHOC-Module, die Bestandteil des virtuellen Dokumentes sind, werden zwar in den Aktionen ausgewählt, die Konfiguration geschieht aber unabhängig von der Generierung der XML-Struktur. D.h. in der XML-Struktur wird vermerkt, dass Module hinzugefügt worden sind, der Inhalt der Module wird in den entsprechenden Modulen selbst erzeugt und verwaltet. Ob hierfür die Datenbank erweitert wird, oder die Informationen anderweitig verwaltet werden, ist Sache des Modul-Entwicklers. Dieses Vorgehen erlaubt neue Module einzubinden, ohne den Erstellungsprozess nach jedem neuen Modul anpassen zu müssen.

Die Module müssen durch den Autor konfiguriert werden. Dazu werden Links zu den Konfigurations-Seiten angeboten. Benutzt der Autor diese Links nicht, so sieht ein späterer Benutzer entweder das Modul in seinem minimalen Zustand (z.B. das Feedback-Modul hat nur Felder für Name, Kommentar und Email-Adresse aber keine weiterführenden Fragen) oder eine Mitteilung, dass dieses Modul durch den Autor nicht konfiguriert worden ist. Wenn immer möglich, sollte ein Modul-Entwickler die erste Lösung ins Auge fassen. Welche Module existieren, wird in der Modul-Konfigurationsdatei festgehalten. Die Selektion der Module und die Links zu deren Konfigurationsseiten sind im Formular integriert worden, das für die Aktionen zuständig ist.

Das letzte Servlet im Prozess überprüft, ob das XML-Dokument alle nötigen Angaben enthält. Ist dies nicht der Fall wird erklärt, welche Angaben noch getätigt werden müssen. Erfüllt das Dokument die DTD, so zeigt das Servlet den Barcode an und übergibt ihn als Bild dem Autor zwecks Einbindung ins Dokument. Zugleich vermerkt es in der Datenbank, dass die ID ab sofort gültig ist.

Der Autor kann die Navigation durch die Komponenten selbst bestimmen. Er kann auch zurück navigieren, um Eingaben zu ändern. Die Servlets sind so geordnet, dass bereits nach Bearbeitung des ersten Formulars eine syntaktisch korrekte XML-Struktur gebildet werden kann.

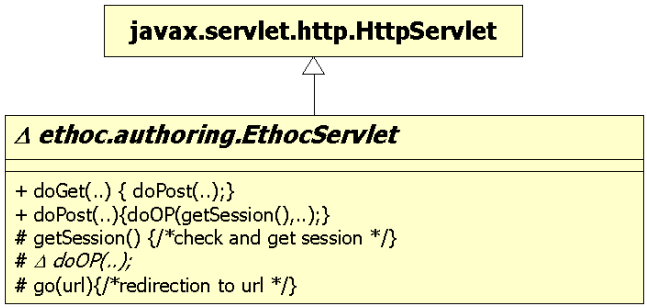

Abbildung 6.4: Methoden des abstrakten ETHOC-Servlets

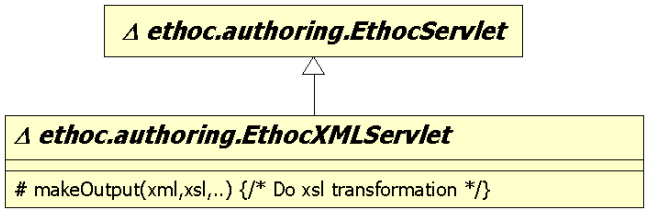

Abbildung 6.5: Methoden des abstrakten ETHOC-XML-Servlets

6.4.3 Abstrakte Servlets

Alle ETHOC-Servlets haben gemeinsame Funktionalitäten. Zum Beispiel muss jedes Servlet prüfen, ob der Autor korrekt authentifiziert worden ist. Dadurch wird verhindert, dass ein Autor sich mittels Manipulation Zugriff zu einem fremden virtuellen Gegenstück verschafft.

Abstraktes ETHOC-Servlet

Das abstrakte ETHOC-Servlet im Paket “Ethoc.Authoring” (siehe Abbildung 6.4) weist konkrete Methoden auf, um die HTTP-Anfrage zu analysieren und zwecks Bearbeitung der Anfrage, entsprechende Funktionen innerhalb des Servlets aufzurufen. Dies geschieht in der Methode “doPost” resp. “doGet”. (“doPost” wird bei einem HTTP-POST-Request und “doGet” bei einem HTTP-GET-Request aufgerufen. Beim ersteren werden allfällige Parameter innerhalb des HTTP-Headers, beim letzteren mit der URL übergeben.)

In diesen Methoden muss in allen Servlets geprüft werden, ob die Session noch gültig ist (Methode “getSession”) und abhängig von den Parametern Operationen ausgeführt werden (Methode “doOP”). “GetSession” ist bereits im ETHOC-Servlet ausprogrammiert. “DoOP bleibt” abstrakt. Die Methode “go” leitet zu einer URL weiter.

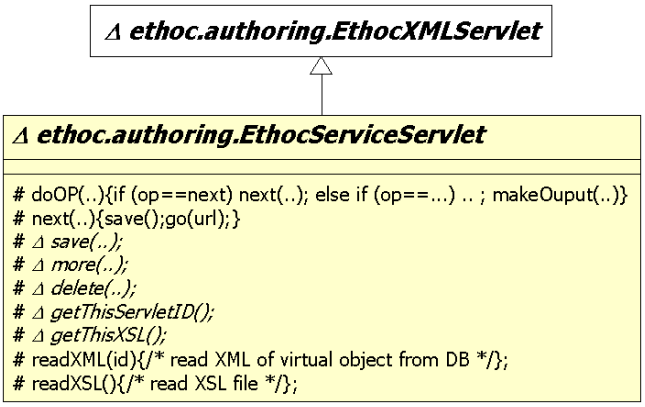

Abbildung 6.6: Methoden des abstrakten ETHOC-Service-Servlets

Abstraktes ETHOC-XML-Servlet

Alle Servlets, welche ihre Ausgaben aufgrund einer XML-Transformation erzeugen, erhalten diese Funktionalität in der abstrakten Klasse EthocXMLServlet (siehe Abbildung 6.5). Diese Funktionalität ist in der Methode "makeOutput" festgehalten, welche die Transformation aufgrund der beiden Parameter durchführt.

Abstraktes ETHOC-Service-Servlet

In den beiden Diensten/Werkzeugen "Registrierung eines neuen Dokumentes" und "Änderungen eines bestehenden Dokumentes" kommen ebenfalls eine Reihe von Servlets zum Zuge. Dies Servlets haben eine starke Gemeinsamkeit: Sie greifen aufs virtuelle Dokument und die Datenhaltung zu. Aus diesem Grund werden alle vom Servlet "EthocServiceServlet" abgeleitet, welches die gemeinsame Funktionalität zur Verfügung stellt (siehe Abbildung 6.6).

Alle Service-Servlets bieten drei Operationen an: Die XML-Struktur erweitern (Methode "more"), reduzieren (Methode "delete") oder zum nächsten Servlet gehen, um einen anderen Bereich der XML-Struktur zu bearbeiten (Methode "next", mit Parameter die Identifikation des Ziel-Servlet). In diesen Operationen werden alle getätigten Änderungen abgespeichert. Aus diesem Grund wird die Methode "doOP" bereits im abstrakten Servlet implementiert.

Ebenfalls implementiert sind die beiden Methoden zum Lesen von XML ("readXML") und XSL ("readXSL"). Erstere liest die XML-Struktur des virtuellen Gegenstückes aus der Datenerhaltung und letztere liest die zum Servlet passende XSL-Datei ein.

6.4.4 Servlet-Hierarchien

Dieser Abschnitt befasst sich mit der Frage, wie die in den vorhergehenden Abschnitten betrachteten Servlets eingebunden und organisiert werden. Es ist

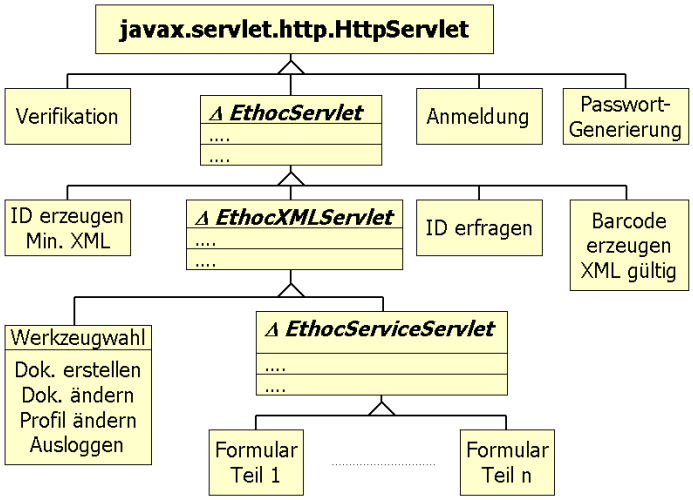

Abbildung 6.7: Servlet-Hierarchie im Autorenbereich

festgestellt worden, dass die Servlets 1 bis n zur Generierung der virtuellen Gegenstücke identische Funktonalität, wie die Servlets 1 bis n zum Update eines virtuellen Gegenstückes bieten. D.h. es können die gleichen Servlet-Klassen verwendet werden. Die Servlets müssen nur wissen, welche Dokumenten-ID gerade bearbeitet wird, und dies ist in der Session festgehalten. Dadurch ergibt sich die in Abbildung 6.7 festgehaltene Servlet-Hierarchie.

Alle Servlets, welche die Session verfolgen müssen, sind eine Ableitung von "EthocServlet". Alle Servlets, die ihre Ausgabe mittels XML und einer XSL-Transformation erstellen, sind Subklassen von "EthocXMLServlet". Wie bereits behandelt, sind alle Servlets, die als XML-Dokument die Struktur des virtuellen Gegenstückes lesen, von "EthocServiceServlet" abgeleitet. Damit jenes Servlet, welches sich um die Anzeige der Auswahl der Module kümmert, nicht speziell behandelt werden muss, wird das Servlet, das den Update-Service initiiert, erweitert. Nachdem es die XML-Struktur eines zu aktualisierenden virtuellen Gegenstückes aus der Datenbank gelesen hat, vergleicht es die Modul-Einträge mit der Konfigurationsdatei. Gibt es neue Module, die nicht in der XML-Struktur des virtuellen Gegenstückes festgehalten sind, werden die neuen Module in die XML-Struktur des virtuellen Gegenstückes eingefügt.

6.5 Datenbank

Viele Freeware-Datenbanken bieten nur eine eingeschränkte Funktionalität. Zum Beispiel fehlen häufig Trigger und/oder Stored Procedures. Aus diesem Grund wird im Design eine Lösung entwickelt, welche diese Tatsache berücksichtigt, um bei der Wahl der Datenbank nicht eingeschränkt zu sein.

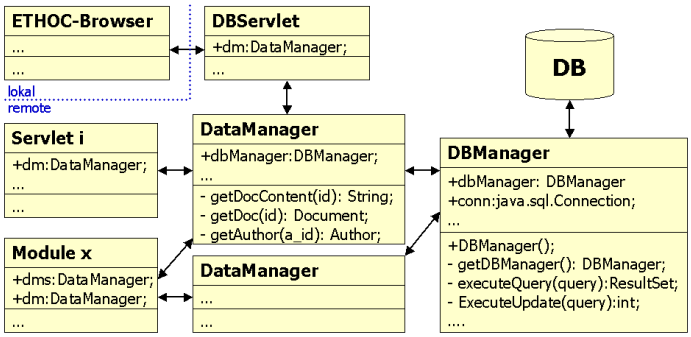

Abbildung 6.8: Zugriff auf Informationen in der Datenbank

6.5.1 Zugriff auf die Datenbank

Die Servlets müssen unabhängig von ihrer Position in der Hierarchie auf die Datenbank zugreifen können. Hinzu kommen Module, die ebenfalls auf ihre eigenen und auf die Daten der virtuellen Gegenstücke zugreifen wollen. Dabei muss die URI der Datenbank bekannt sein, sowie die technische Benutzer-ID[1] und das dazugehörende Passwort. Auch auf Benutzerseite muss auf die Datenbank zugegriffen werden. Die Funktionalität des Datenbankzugriffes gilt es abzukapseln, damit nicht in jedem Programm das Passwort zum Datenbankzugriff bekannt sein muss und damit nicht ständig der gleiche Kode geschrieben werden muss.

Die Abkapselung geschieht über mehrere Stufen (siehe Abbildung 6.8). Auf die Datenbank direkt greift nur der "DBManager" – der Datenbank-Manager – zu. Er verwaltet eine Verbindung zur Datenbank und bietet eine einfache Schnittstelle, um auf der Datenbank zu arbeiten. Dies geschieht über die Abfragesprache der Datenbank, z.B. SQL. Der DB-Manager ist als Singleton implementiert. Dadurch wird sichergestellt, dass in jeder ETHOC-Installation diese Klasse einmalig existiert. Somit kann z.B. später zwecks Leistungssteigerung ein Cache-Management im DB-Manager eingeführt werden. Der Einsatz eines Cache dürfte hier zu einem Performanzgewinn führen, da anzunehmen ist, dass ein Autor, der gerade eine XML-Struktur bearbeitet, diese regelmässig benötigt. Dadurch erspart sich das System zeitintensive Datenbankanfragen. Auf die Aspekte der Verteiltheit wird in Abschnitt 7.8 genauer eingegangen.

Auf den DB-Manager greifen ausschliesslich Daten-Manager ("DataManager") zu. Von diesen kann es mehrer Implementierungen geben. So können Module auf ihre Daten über ihre eigenen Daten-Manager zugreifen. Wollen sie auf Daten, die das virtuelle Gegenstück betreffen zugreifen, so bedienen sie sich des Daten-Managers im Authoring-Paket. Die Aufgabe eines Daten-Managers ist es die Abfragesprache der Datenbank abzukapseln. Über logische Methoden wie "getDoc(id)", "getDocContent(id)" oder "getAuthor(authorID)" operiert das ETHOC-System mit dem Daten-Manager und erhält die Daten in einem Datenbank-unabhängigen Format, wie "Ethoc.System.Doc", "java.lang.String"

[1] Eine technische Benutzer-ID wird nicht von Personen, sondern von Programmen resp. Prozessen benutzt. Auf die ETHOC-Datenbank greifen die Servlets mit einer speziellen – eben der technischen – Benutzer-ID und nicht mit der jeweiligen ID der Autoren zu.

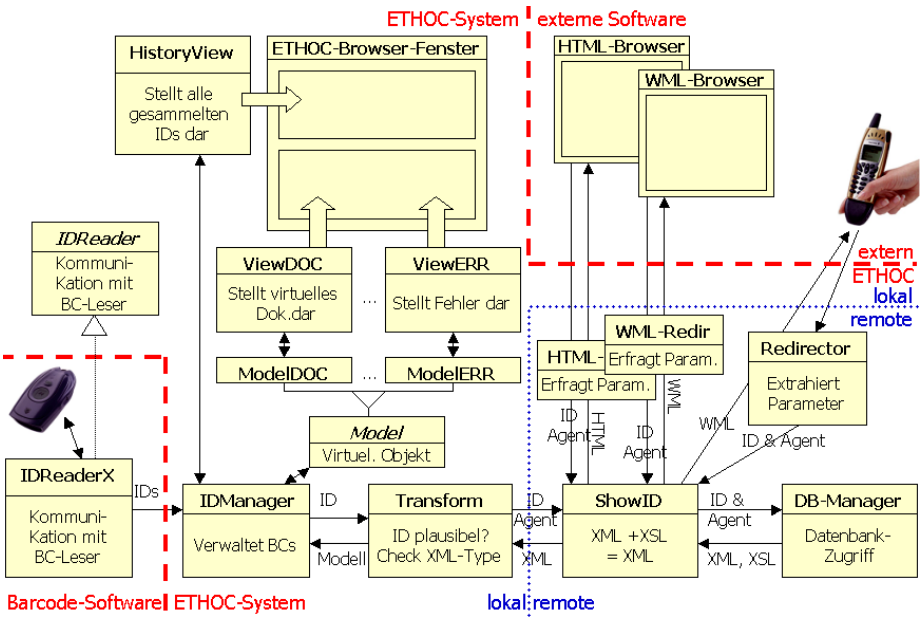

Abbildung 6.9: Benutzungsmöglichkeiten des ETHOC-Systems

oder “Ethoc.System.Author”. Der Daten-Manager organisiert also die Daten in der gewünschten Form (Java-Objekte, XML etc.). Dadurch kann später die Datenbank ausgetauscht werden, ohne dass das ETHOC-System von der Änderung betroffen ist.

Ein kleines Beispiel um die Aufteilung näher zu erläutern: Die “getDoc(id)”-Methode im “DataManager” generiert eine SQL-Query mit welcher die Methode “executeQuery(query)” im DB-Manager aufgerufen wird. Der DB-Manager liefert ein “ResultSet” aus dem Paket “java.sql”, welches das gewünschte Resultat beinhaltet. Der “DatenManager” extrahiert die Daten aus dem “ResultSet” und benutzt diese, um ein “Ethoc.System.Doc”-Objekt zu generieren, welches er als Resultat zurückgibt.

Der Ethoc-Browser greift über das Servlet “DBServlet” auf den DataManager zu. Wäre der DB-Manager Teil des Ethoc-Browsers, so könnte ein versierter Benutzer leicht eigene Java-Programme schreiben, die direkt auf den DB-Manager zugreifen und über diesen beliebige SQL-Befehle ausführen. Ebenfalls könnte durch eine Dekompilierung das Passwort des technischen Benutzers herausgefunden werden und somit direkt auf der Datenbank gearbeitet werden.

6.6 Benutzerseite

Für den Benutzer gibt es vier Wege, um an ein virtuelles Dokument zu gelangen (siehe Abbildung 6.9). Der erste Weg (in der Abbildung unten links) ist via Barcode-Leser und dem lokal installierten ETHOC-Browser. Die Wege zwei und drei (in der Abbildung oben rechts) führen über einen Browser ohne vorhande-

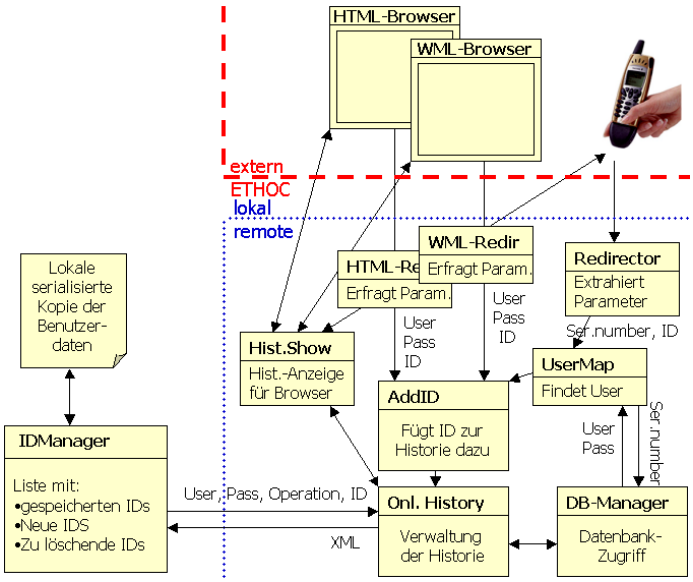

Abbildung 6.10: Umgang mit der zentralen Historie

nen Barcode-Leser. Die ID wird dabei von Hand eingegeben. Der vierte Weg (in der Abbildung oben rechts) führt über einen AirClic-Leser am Mobiltelefon.

6.6.1 Lokale Komponenten

Der ETHOC-Browser besteht aus den in Abbildung 6.9 dargestellten lokalen Komponente, die in diesem Abschnitt näher erläutert werden.

ID-Reader

Für jeden Barcode-Leser-Typ muss ein Treiber (In Abbildung 6.9 als “IDReaderX” bezeichnet) geschrieben werden. Dieser erweitert die Schnittstelle “IDReader”, kommuniziert mit dem Barcode-Leser und übergibt dem lokalen ID-Manager die IDs.

ID-Manager

Der ID-Manager synchronisiert sich, sofern eine Internet-Verbindung vorhanden ist, mit der Online-Historie (siehe Abbildung 6.10). Dabei wird festgestellt, welche IDs von den Autoren aktualisiert worden sind. Die neuen IDs werden in die Verwaltung aufgenommen und aufgelöst. Die IDs der Verwaltung können von allen lokalen Komponenten und dem Benutzer ausgetauscht und gelöscht werden. Bei Veränderung der lokalen Historie wird die Online-Historie nachführt.

Model-Objekte

Ein Model-Objekt ist die Repräsentation der XML-Struktur eines virtuellen Dokumentes in Form eines Java-Objekts. Ein Model kennt seine ID und seine XML-Struktur und hat eine Referenz zu einer View-Klasse, die es darstellt.

Ein Model-Objekt hat Zugriff auf die lokale Historie. Dadurch sind Objekte denkbar, die mit der Historie arbeiten. Zum Beispiel könnte so ein Drucker-Model automatisch das zuvor eingelesene Dokument ausdrucken. Das Model-Objekt wird wie zuvor beschrieben, in die Verwaltung aufgenommen und zeigt im oberen Teil des ETHOC-Browser-Fensters unter Zuhilfename seiner View-Klasse eine Vorschau seines Inhaltes. Wird das Objekt selektiert, so ruft es eine weitere Methode seiner View-Klasse auf und veranlasst seine vollständige Ausgabe im unteren Teil des ETHOC-Browser-Fensters.

HistoryView

Die "HistoryView" stellt die Verwaltung grafisch dar. Jede ID beansprucht eine Zeile. Es können IDs entfernt und alle IDs nach neuen Inhalten prüft werden. Entfernt der Benutzer eine ID, so wird sie auch in der Online-Historie gelöscht. Ist gerade keine Online-Verbindung vorhanden, so werden die zu löschenden IDs bis zur nächsten Verbindung speziell verwaltet.

Transformator

Jede neue ID wird mit Hilfe der Klasse "Transform" aufgelöst. Wird eine ID bereits verwaltet, so wird das Model-Objekt mit der selben ID befragt, ob es sinnvoll ist, mehrfach verwaltet zu werden. Ist dies der Fall, so wird das Objekt dupliziert; andernfalls, wie z.B. bei Dokumenten, wird das Datum des letzten Einlesens aktualisiert.

Die Klasse Transform organisiert die XML-Repräsentation der ETHOC-ID. Aus der XML-Antwort generiert sie ein passendes Model-Objekt. Ist das Typ-Attribut des ETHOC-Elementes – das erste Tag aller virtuellen Dokumente – gleich der Zeichenkette "doc" so handelt es sich um ein Dokument, und es kann ein Dokumenten-Model generiert werden. Fehlermeldungen vom Server erhalten als Typ-Attribut den Wert "err". Die Klasse "Transform" wird also benötigt, um aus IDs Model-Objekte zu generieren. Welcher XML-Typ welche Model-Klasse entspricht, wird anhand von Namenskonvention gelöst.

6.6.2 Online-Komponenten

"ShowID" wird bei allen vier Zugriffsmöglichkeiten immer kontaktiert (siehe Abbildung 6.9). Beim ersten Weg geschieht dies über die Transform-Klasse. Im Weg zwei und drei wird die Adresse der Redirector-Klassen über einen Browser angewählt. Diese fragen beim Benutzer nach der ETHOC-ID. Gibt der Benutzer gleich seinen Benutzernamen und das Passwort an (evtl. sind diese Angaben bereits in Cookies oder in anderer Form gespeichert, dann entfällt die wiederholte Eingabe), hat er auch Zugriff auf seine Historie. Mit der ID kontaktieren sie

"ShowID". Im Hintergrund sorgen die Redirector-Klassen dafür, dass die soeben aufgelöste ID auch in die Historie eingetragen wird (siehe Abbildung 6.10).

In der HTTP-Anfrage ist immer der "user agent" angegeben, d.h. es kann auf den Browser-Typ geschlossen werden, sofern dieser bekannt ist. Da die Benutzer aber entweder über die Redirectoren-Klassen oder den Transformator zu "ShowID" gelangen, ist bereits bekannt, ob die Klienten-Applikation HTML, WML oder XML versteht. Für jeden ETHOC-Typ wird deshalb für HTML, WML und XML eine eine XSLT-Datei in der Datenbank hinterlegt. Die Antwort des Servlets "ShowID" ist eine Transformation der XML-Struktur des zur ID gehörenden virtuellen Gegenstückes mit einem XSLT-Dokument, welches zum virtuellen Gegenstück und zur Klienten-Applikation passt. Im Fall eines WML-Browsers ist es WML, im Fall des HTML-Browsers ist es HTML und im Fall der Transformer-Klasse ist es ein XML-Dokument.

Auch der vierte Weg (das Benutzen eines AirClicLesers) führt über "ShowID". Der Benutzer wird zum "Redirector" umgeleitet. Dieser extrahiert die Barcode-ID und die AirClic-Benutzerkennung und leitet die Anfrage mit der ID als Parameter an "ShowID". weiter. Auf die besprochen Art und Weise wird eine WML-Datei generiert, welche ans Mobiltelefon retourniert wird. Auch der Redirector von AirClic sorgt im Hintergrund dafür, dass die ID in die Historie hinzugefügt wird. Anhand der Serienummer des AirClic-Barcode-Lesers kann bei der Datenhaltung der Benutzername und das Passwort erfragt werden. Mit diesen Angaben wird die ID hinzugefügt. Die Sicherheitsprobleme, die diese Lösung mit sich bringt, wird im Abschnitt 7.9 behandelt.

Um die Historie zu besichtigen, bietet jedes virtuelle Dokument, das zu WML oder HTML transformiert worden ist, einen Link zum Servlet "HistoryShow". Gegebenenfalls muss der Benutzer sich wieder einloggen. Jeder Eintrag wird als Link angezeigt, der zur Ansicht des virtuellen Gegenstückes führt. Bei der Darstellung als WML muss darauf geachtet werden, dass die maximale Grösse eines Decks beschränkt ist. Diese Beschränkung variert von Gerätehersteller zu Gerätehersteller. Laut [30] beträgt diese Grösse bei einem Nokia 7110 1397 Byte und bei einem Ericsson R320 3000 Byte. Aus diesem Grund wird empfohlen, das Limit von 1.3 Kilobyte nicht zu überschreiten. Es empfiehlt sich aber auch von der ergonomischen Seite aus, nicht mehr als zehn IDs auf einmal anzuzeigen. Ähnlich wie bei Suchmaschinen, werden Links zu den nächsten und vorherigen zehn IDs führen.

Die Aufgabe des DB-Managers wurde in Abschnitt 6.8 behandelt.

Kapitel 7

Implementierung

In diesem Kapitel wird die Implementierung näher erläutert. Insbesondere werden spezielle Konzepte, Schnittstellen und Strukturen hervorgehoben, die im Design nur kurz behandelt oder offen gelassen worden sind. Des weiteren werden Schwierigkeiten erläutert, die mit den verwendeten Bibliotheken aufgetreten sind.

7.1 XML-Strukturen

Dieser Abschnitt erklärt den Aufbau der XML-Strukturen in ETHOC. Die Document Type Definition der Strukturen ist im Anhang D vollständig spezifiziert.

7.1.1 Virtuelle Gegenstücke

Die Struktur der virtuellen Gegenstücke beginnt mit dem Element (XML-Tag) “ethoc”. Es besitzt ein Attribut “type”, welches bei Dokumenten den Wert “doc” haben muss. Wird das ETHOC-System später um andere Typen erweitert, so wird eine neue Struktur benötigt, welche mit dem Element “ethoc” beginnt und im Attribut “type” den neuen Typ aufweist. Innerhalb des Elements “ethoc” befindet sich das Element “virt_doc” mit den Attributen “id” – die ID des Dokumentes und “last_change” – das Datum der letzten Änderung. Innerhalb des Elemtens virt_doc finden die Elemente “author”, “doc_info”, “contacts” und “extend” ihren Platz. “Extend” sind die Aktionen des virtuellen Dokumentes.

Einige Elemente verfügen über das Attribut “last_change”. Dadurch kann schnell herausgefunden werden, wann welche Komponenten geändert worden sind. Es ist somit möglich, einen Benutzer genau zu informieren, was sich seit seinem letzten Kontakt verändert hat, sofern eine ältere XML-Struktur zum Vergleich vorhanden ist. Überall, wo kein Attribut “last_change” vorhanden ist, wird auf die Lokalisierung des Updates verzichtet. Sie ist aber dennoch möglich, da zwei XML-Strukturen verglichen werden können. Bei den Modulen ist via Attribut “last_config” vermerkt, welche Version die Konfigurationsdatei hatte, als das virtuelle Gegenstück erstellt wurde. Ist die Versionsnummer der Module des

virtuellen Dokumentes von der Nummer der aktuellen Konfigurationsdatei verschieden, so gibt es neue Module. Die neuen Module werden eingefügt und der Autor wird beim Bearbeiten seines Dokumentes die neuen Module zur Auswahl vorfinden.

Diese XML-Struktur wird verwendet, um zusammen mit einem XSLT-Dokument die HTML-Formulare zu generieren, in welchen ein Autor seine Daten ändern kann. Mit weiteren XSLT-Dateien werden eine Übersicht für den Autor, eine HTML-Ansicht für die Internet-Benutzer, eine WML-Ansicht für Mobiltelefonbenutzer und eine XML-Struktur für den ETHOC-Browser erstellt.

7.1.2 Modulkonfiguration

In der XML-basierten Modulkonfiguration werden alle Module mit dem Zeitpunkt ihres individuellen Einbindens in ETHOC (Version der Konfigurationsdatei zu diesem Zeitpunkt) und weiteren Angaben festgehalten. Das Wurzelement heisst “modules”. Sein Attribut “last_config” ist die aktuelle Versionsnummer der Konfigurationsdatei. Innerhalb des Wurzelelementes befinden sich einzelne Elemente namens “module”. Als Attribute erhalten sie den Einfügezeitpunkt und die Email-Adresse der Person, die das betreffende Modul eingeführt hat. Zudem ist angegeben, ob das Modul standardmässig aktiviert oder deaktiviert sein soll. Jedes Modul hat einen Eintrag “name”, “label” und “entry_point_config”. Der Name identifiziert das Modul innerhalb von ETHOC und muss eineindeutig sein, das Label dient zur Benennung des Modules gegenüber dem Benutzer und “entry_point_config” ist eine URL – vorzugsweise die Adresse eines Servlets – welche als Einstiegspunkt zur Konfiguration des Moduls durch den Autor dient. Die Elemente “entry_point_use” und “notifier” sind optional. Erstes ist die Internet-Adresse (auch hier üblicherweise die URL eines Serlvets), an welche der Benutzer umgeleitet wird, wenn er das Modul auswählt. Letztere ist eine Java-Klasse, welche durch das ETHOC-System aufgerufen wird, wenn virtuelle Gegenstücke erzeugt oder geändert werden. Dadurch bekommt ein Modul alle Änderungen mit. Die unter “notifier” aufgeführte Klasse implementiert das Interface “Ethoc.System.ModuleNotifier”.

7.1.3 Autoren-Werkzeuge

Die Anzahl Autoren-Werkzeuge ist ebenfalls flexibel. Um späteren Entwicklern eine leichte Verwaltung der Werkzeuge zu ermöglichen, werden alle vorhandenen Autoren-Werkzeuge in einer XML-Datei aufgelistet. Diese Datei wird auch verwendet, um dem Autor ein Auswahl-Menü der Werkzeuge anzuzeigen. Innerhalb des Wurzel-Elements “authoring” befinden sich Elemente namens “tool”, die ein Werkzeug umfassen. Darin wird die Text-Anzeige an den Autor spezifiziert und das erste Servlet, welches den Eingabeprozess beginnt, angegeben.

7.2 Autorenportal

7.2.1 Einstiegsseite

Das Autorenportal wird vom Autor mittels eines HTML-Browsers bedient. Nicht alle Browser sind gleich mächtig. Versierte Internetbenutzer schalten oft Funktionen des Browsers ab, weil sie mit diesen Fuktionen schlechte Erfahrungen gemacht haben oder die Befürchtung haben, ihre Privatsphäre könnte durch das Eingeschaltetlassen ausspioniert werden.

ETHOC benutzt Sessions, um sich zu merken, welcher Autor gerade welches Dokument bearbeitet. Der von der Java-Servlet-API zur Verfügung gestellte Mechanismus benutzt dazu Cookies. Die Navigation und die Fehlerüberprüfung nutzen JavaScript. um ETHOC benutzen zu können, müssen demzufolge Cookies und JavaScript eingeschaltet sein. Die Aufgabe der Einstiegsseite ist, dies zu überprüfen. Unterstützt der Browser die beiden Funktionalitäten, so wird er zu einem Login-Fenster umgeleitet. Ist dies nicht der Fall, so wird ein Hinweis, dass die entsprechende Funktionalität im Browser aktiviert werden sollte, um ETHOC benutzen zu können, angezeigt. Um unerfahrenen Benutzer allfällige Vorbehalte betreffend JavaScript oder Cookies zu nehmen, werden Links mit weiteren Informationen zu JavaScript und Cookies aufgeführt.

Es wird in diesem Projekt davon ausgegangen, dass ein Autor nicht mitten in der Bedienung die Konfiguration des Browsers ändert. Tut er dies, indem er z.B. JavaScript ausschaltet, so funktioniert die Navigation nicht mehr und er kann nicht mehr weiternavigieren, sprich keine Daten speichern. Schaltet er Cookies ab, so finden die Servlets die Session nicht mehr und reagieren mit einer Exception. Ebenfalls wird davon ausgegangen, dass der Benutzer nicht böswillig ist und versucht mit selbst generierten Formularen unerlaubte Eingaben zu erreichen. Diese Annahme ist deshalb gerechtfertigt, weil ein Autor dadurch dennoch nur seine eigenen Dokumente verändern könnte.

7.2.2 Autoren-Identifikation

Die Autoren-Identifikation ist wie im Design (siehe Abschnitt 6.4.1) beschrieben, realisiert worden. Beim Service für “vergessliche” Benutzer wurde nebst der Passwortgenerierung, die Möglichkeit geschaffen, auch die Autoren-ID anhand der Email-Adresse(n) zu erfragen.

7.2.3 Virtuelle Gegenstücke erstellen und ändern

Ein Autor muss maximal vier Formularseiten bearbeiten, in denen Angaben zum Dokument (wie Titel und Gültigkeit), zum Autor, zu den Kontaktpersonen und zu den Aktionen gemacht werden. Nicht im Design vorgesehen aber implementiert worden, ist eine Abschlussseite, die das ganze Dokument anzeigt und den Autor ausloggen oder andere Werkzeuge wählen lässt. Ein neues Dokument ist bereits nach dem ersten Servlet gültig, d.h. es kann sofort der Barcode bezogen und dann ausgeloggt werden. Die Autorennotifikation ist als Modul, das standardmässig aktiviert ist und ohne Konfiguration den Autor sieben Tage

und ein zweites Mal einen Tag vor Ablauf der Gültigkeitsperiode benachrichtigt, realisiert worden. Das Feedback-Formular ist standardmässig deaktiviert. Ohne Konfiguration wird ein minimales Formular generiert, das einen Standard-Begrüssungstext und Felder für Name, Vorname, Email-Adresse und Kommentar des Benutzers enthält.

7.2.4 Datei-Upload

Eine zu Beginn nicht erwartete Schwierigkeit stellte sich mit dem Datei-Upload. Um Dateien via HTTP zum Server hochzuladen, erhält das "form"-Element in einer HTML-Seite das Attribut "enctype" mit dem Wert "multipart/form-data". Dadurch aber werden alle Parameter vollständig neu kodiert übergeben und die Get-Methoden in "ServletRequest", mit welchen normalerweise auf Servlet-Seite die Parameter extrahiert werden, finden keine Parameter mehr. Es muss entweder der Request selbst geparst werden, indem der Standard analysiert und befolgt wird, oder ein Java-Paket verwenden werden, dass diese Aufgabe übernimmt. In dieser Arbeit wurde letzteres gewählt und die Bibliothek COS [21] verwendet. Es wurde ein Wrapper geschrieben, der einige Funktionen eines "ServletRequest" anbietet, aber im Hintergrund COS benutzt. Aufgrund eines Fehlverhaltens von COS musste die Klasse dennoch leicht umgeschrieben werden (siehe Anhang C.8). Das Verwenden dieses Paketes hat auch zur Folge, dass der Header maximal 1GB gross sein darf. Im Header befinden sich die Namen der Parameter und deren Werte, sowie Metadaten zu den hochgeladenen Dateien inklusive des ganzen Dateiinhalts.

7.2.5 Barcode-Erzeugung

Barcodes werden im JPG- und GIF-Format erzeugt. Um JPGs zu erzeugen, wird das Paket "com.sun.image.codec.jpeg" [22] verwendet, welches schon standardmässig in JDK 1.2 vorhanden ist. Um GIFs zu erzeugen, wurde Acme Labs GIFEncoder [23] verwendet.

7.3 Module

Die Module werden beim Start des ETHOC-Systems dynamisch eingebunden (siehe Abschnitt 7.1.2). Sie besitzen zwei Einstiegspunkte. Einen zur Konfiguration des Moduls durch den Autor und einen zur Bedienung durch den Benutzer. Die Autoren-Notifikation hat keinen Einstiegspunkt für den Benutzer. D.h. es ist ein Modul, das der Benutzer nicht wahrnimmt. Das Feedback-Formular hat hingegen einen Einstiegspunkt für den Benutzer, um diesen zu weiteren Servlets, die eine kliententypische Ausgabe, wie HTML und WML erzeugen, umzuleiten.

Beide Module benutzen resp. erweitern Klassen, welche die ETHOC-Infrastruktur zur Verfügung stellt. Die Einstiegspunkte für den Autor sind Erweiterungen der Klasse "Ethoc.Authoring.EthocServlet". Der Einstiegpunkt für Benutzer bedient sich der Klasse "Ethoc.System.AgentChecker", um den Klienten weiterzuleiten. Diese Klasse bestimmt anhand des User-Agents, ob der Browser WML oder HTML versteht.

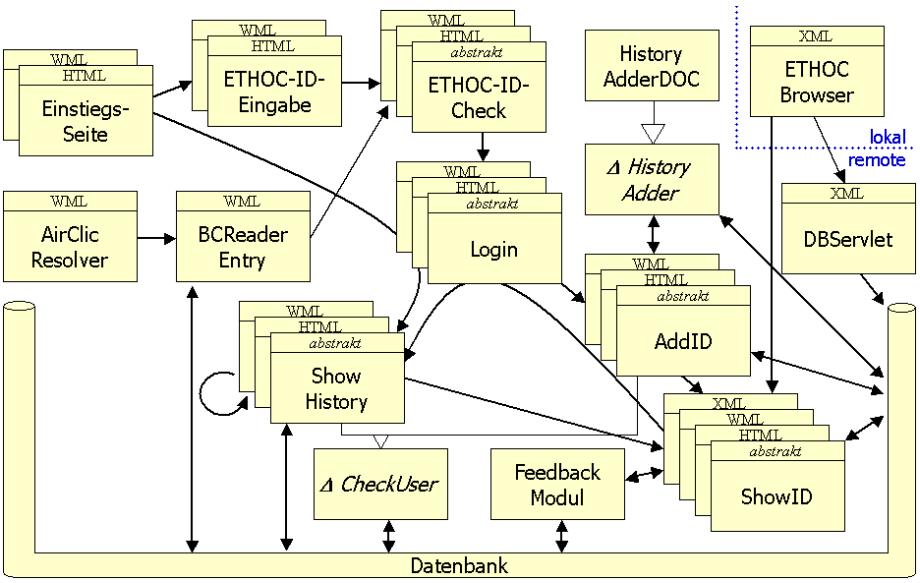

Abbildung 7.1: Klassen und Navigation im Benutzerportal

Über Änderungen an virtuellen Gegenstücken wird ein Modul durch seinen Modul-Notifier, welcher das Interface "Ethoc.System.ModuleNotifier" implementiert, informiert. Das ETHOC-System verwaltet die Notifier aller Module. Tritt eine Aktion ein, die ein bestimmtes Dokument oder ein Modul betrifft, ruft es Methoden, die im Interface "Ethoc.System.ModuleNotifier" definiert sind, auf. Auf diese Weise kann jedes Modul individuell reagieren und eigene Aktionen veranlassen. Zum Beispiel passt das Autoren-Notifikations-Modul die Notifikationsdaten an, wenn die Gültigkeit eines Dokumentes verändert wird.

7.4 Email-Verkehr

Das ETHOC-System sendet Emails an die Autoren und an die Benutzer, z.B. um die Passworte zu übermitteln oder als Funktionalität in den Modulen. Hierfür wird die Klasse "sun.net.smtp.SmtpClient", mit "ethoc@ethz.ch" als Absenderadresse verwendet. Diese Klasse genügt den momentanen Anforderungen von ETHOC, ist aber keine Standardbibliothek. Es wird von Sun nicht garantiert, dass es die Bibliothek auch in späteren Versionen in dieser Form gibt. Sollen später auch Attachements versendet werden, so muss ein mächtigeres Paket wie z.B. Suns JavaMail [24] verwendet werden. Dies kann einfach durchgeführt werden, da die Funktionalität des Email-Versendens durch eine ETHOC-eigene Klasse abgekapselt wird.

7.5 Benutzerportal

In diesem Anschnitt wird das Benutzerportal – die Online-Komponenten der Benutzerseite, detailierter als im Design-Abschnitt 6.6.2 beschrieben. In Abbildung 7.1 sind die an einer Abfrage beteiligten Java-Klassen abgebildet. Der Benutzer hat von der Einstiegsseite aus die Wahl, entweder seine Online-Historie zu betrachten resp. zu bearbeiten oder eine neue ETHOC-ID anzuschauen. Es gibt für HTML und WML separate Einstiegsseiten. Das System erfährt somit, ob der Browser WML oder HTML versteht. Die weitere Navigation erfolgt durch Klassen, welche die gewünschte Ausgaben erzeugen. Meist sind die Klassen so organisiert, dass abstrakte Klassen die Funktionalität sicherstellen und die konkreten Klassen die Ausgabe darstellen.

Entscheidet sich ein Benutzer für eine neue ID, so wird er vom ID-Eingabe-Servlet aufgefordert, die ID von Hand einzugeben. Die eingegebene ID wird im nächsten Servlet auf ihre Gültigkeit geprüft. Ist die ID korrekt, so wird der Benutzer aufgefordert sich einzuloggen, sofern er die ID in seine Historie einfügen will. "AddID" prüft aufgrund der Ableitung von "CheckUser" ob der Benutzer korrekt eingeloggt ist und überlässt das Einfügen der ID in die Benutzerhistorie der Klasse "HistoryAdder". Ist dies geschehen, wird der Benutzer zum Servlet "ShowID" weitergeleitet, welches die ETHOC-ID anzeigt. Hierfür wird eine XSL-Transfomation durchgeführt. Bei WML und HTML hat es einen Link zur persönlichen Historie des Benutzers. Entscheidet sich ein Benutzer, seine Historie zu betrachten, so wird er über den Login zur Historie weitergeleitet. Sind ID und Passwort nicht gespeichert (in HTML werden Cookies benutzt und in WML Variablen), muss er sich erneut einloggen. Auf Seiten der Benutzer wird ein anderer Weg beschritten als auf Seiten der Autoren. Das Benutzer-Portal kann auch ohne Cookies und Java-Script benutzt werden. Aus diesem Grund kann keine Session verwendet werden. Die Benutzerkennung ohne Cookies ist lediglich bis zum nächsten Servlet nach dem Login verfügbar. Wird die Benutzer-Authentifikation in einem Modul benötigt (z.B. in einem späteren Abstimmungsmodul) so muss der Benutzer über den Login geführt werden. Im Feedback-Modul wurde darauf verzichtet, weil auch Benutzer ohne Historie das Modul benutzen können sollen.

Ein weiterer Einstiegspunkt im Benutzer-Portal sind auf Barcode-Leser ausgerichtete Servlets. Der "AirClicResolver" nimmt die Anfrage so entgegen, wie von der Firma AirClic spezifiziert und wandelt sie in eine geräteunabhängige Anfrage um, welche zum Servlet "BCReaderEntryWML" weitergereicht wird. Jenes Servlet organisert anhand Barcode-Leser-Typ und Identifizierung (im Falle von AirClic die Serienummer des Lesers) ID und Passwort des Benutzers und speichert die Daten als WML-Variablen. Dadurch muss der Benutzer sich nicht ausweisen. Um andere Leser ans ETHOC-Benutzer-Portal anzuschliessen müsste pro Gerät eine Klasse der Art "AirClicResolver" geschrieben werden. Das Ermitteln der Benutzerdaten anhand der Seriennummer eines Barcode-Lesers birgt potentielle Sicherheitsrisiken (siehe Abschnitt 7.9). Wünschenswert wäre, wenn der Barcode-Leser weitere Angaben zur Authentifizierung übermitteln könnte. Dies ist aber bei AirClic nicht der Fall. Für mächtigere Barcode-Leser wird empfohlen, einen anderen Einstiegspunkt zu programmieren oder den Weg über den lokalen ETHOC-Browser zu verwenden.

Der ETHOC-Browser kontaktiert das "DBServlet" um sich mit der Historie zu

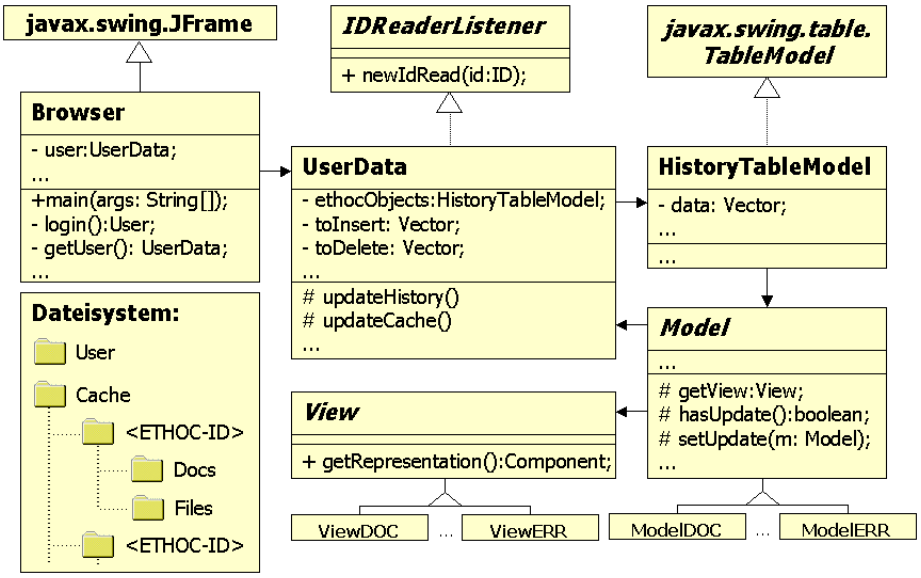

Abbildung 7.2: Hauptklassen des ETHOC-Browsers

synchronisieren. Benötigt er die XML-Struktur einer ID, so wendet er sich direkt an "ShowIDXML".

7.6 ETHOC-Browser

Der ETHOC-Browser wird unter Windows mit einer Batch-Datei gestartet. Es wird die Main-Methode der Klasse "Browser" aufgerufen. Diese erstellt eine neue Instanz "Browser". Als erstes wird ein Fenster angezeigt in welchem sich der Benutzer einzuloggen hat. Ist dies geschehen, werden die Daten online verifiziert. Existiert keine Verbindung ins Internet, werden die Daten im lokalen Profil des Benutzers verifiziert, sofern eines existiert. Das lokale Profil der Benutzer befindet sich im Ordner "User" und besteht aus einem serialisierten "UserData"-Objekt. Als Dateinamen wird die Benutzer-ID mit der Endung ".dat" verwendet. War der Login erfolgreich, wird die Datei in ein Objekt zurücktransformiert und mit dem Browser referenziert. In einem "UserData"-Objekt sind die Präferenzen eines Benutzer und seine Historie abgespeichert. Die Historie besteht aus einem "HistoryTableModel", welches in einem Daten-Vektor Objekte vom Typ "Model" verwaltet. Diese können Aufgrund ihrer View im ETHOC-Browser dargestellt werden. Ein genau Erklärung der Model-View-Idee ist in Abschnitt 6.6.1 vorzufinden. Model-Objekte erhalten eine Referenz zum "UserData"-Objekt (siehe Abbildung 7.2), um die Benutzerdaten bei Bedarf ändern zu können.

Der Dateien-Cache ist für alle Benutzer gemeinsam organisiert. D.h. lädt ein Benutzer Dateien einer ETHOC-ID in den Cache herunter, so stehen sie den anderen Benutzern ebenfalls zur Verfügung und müssen nicht erneut heruntergeladen werden. Pro ETHOC-ID wird innerhalb des Ordners "Cache" ein Ordner

erstellt, der die Nummer der ETHOC-ID trägt. Innerhalb dieses Ordners hat es die Unterordner "Docs" und "Files", in welche die Dokument-Kopien und die zusätzlichen Dateien gespeichert werden. Die Dateinamen bestehen aus einem Zeitstempel dem "@"-Zeichen und dem Dateinamen, den der Autor gewählt hat. Dadurch kann der Browser herausfinden, ob es auf dem Server mittlerweile aktuellere Versionen von Dateien gibt. Die Dateien sind im Cache mit einem Schreibschutz abgelegt. Sie können nur lesend geöffnet werden. Der Benutzer kann wählen, ob er Dateien anschauen oder speichern will. Soll eine Datei geöffnet werden, wird sie direkt aus dem Cache geöffnet. Will der Benutzer die Datei speichern, wird die Datei von Cache kopiert, wobei als Dateiname der vom Autor gewünschte Name als Vorschlag erscheint. In beiden Fällen wird eine Datei zuerst in den Cache heruntergeladen, falls sie dort nicht vorhanden ist.

Im ETHOC-Browser wird der Web-Browser benötigt, um Dateien oder Links zu öffnen. Doch das Aufrufen des Browsers ist betriebssystemabhängig. Es wurde eine Klasse verwendet, die je nach zugrundeliegendem Betriebssystem und Standard-Browser den passenden Startbefehl ausführt. Dieses Vorgehen hat aber den Nachteil, dass wenn das System nicht erkannt werden kann oder der Browser nicht existiert, das Öffnen von Dateien und Links nicht funktioniert. Aus diesem Grund werden im ETHOC-Browser auch die URLs angezeigt und die Dateien können gespeichert werden, um später von Hand geöffnet zu werden.

7.7 Datenbank

Es wird die Open-Source-Datenbank MySQL [27] benutzt. In der aktuellen Version (3.32.46a) werden leider keine Triggers und Stored Procedures unterstützt. Sie sind erst für Version 4.1 angekündigt. Auf MySQL wird über JDBC zugegriffen. Hierfür wird der MM.MySQL-Treiber [29] in der Version 2.0.8 benutzt. Dieser Abschnitt erklärt, wie die Informationen in der Datenbank abgespeichert werden. Es wird ebenfalls darauf eingegangen, was die Einschränkungen der Datenbank sind.

Die vergebenen Koordinator-IDs (der Aufbau der ETHOC-ID wird im Abschnitt 5.2.4 erklärt) werden in der Tabelle "IdCoord" festgehalten (siehe Tabelle 7.1). Im Feld "host" steht die IP-Adresse des ETHOC-Servers und in "id" die an ihn vergebenen Koordinatoren-IDs. Jeweils die grösste ID eines ETHOC-Hosts ist seine aktuelle Koordinatoren-ID. In "last_id" steht die zuletzt vergebene ID innerhalb des Kontextes der Koordinator-ID. Ein ID-Koordinator hat in der momentanen Version keine eigene persistente Speicherung der vergebenen IDs, deshalb muss er bei jeder ID-Vergabe die letzte ID in der Datenbank nachführen. Es ist auch Sache des ID-Koordinators zu überprüfen, ob sein ID-Vorrat erschöpft ist. Die Datenbank ist nicht auf eine fixe ID-Stelligkeit beschränkt. Wird später die ID-Länge vergrössert tangiert es die Datenbank nicht, resp. nur dann, wenn die ID das Speichervermögen einer INT überschreitet (siehe MySQL-Manual [28]).

Die Angaben, die ein Autor bei der Registrierung angeben hat, sind in der Tabelle "Author" festgehalten. Bei der ID und dem Passwort des Autors ist die Gross-/Kleinschreibweise wichtig. Der DB-Manager verschlüsselt das Passwort

Field	Type	Null	Key	Default
host	varchar(16)		PRI	
id	int(11)		PRI	0
last_id	int(11)	YES		NULL

Tabelle 7.1: Definition der MySQL-Tabelle "IdCoord"

mit der MySQL-eigenen Einwegfunktion "PASSWORD". D.h. der Test, ob ein eingegebenes Passwort korrekt ist, geschieht in der Datenbank.

Die virtuellen Gegenstücke werden in der Tabelle namens "Doc" gespeichert. Häufig benötigte Daten der XML-Struktur, wie z.B. Titel und Autor, sind zusätzlich redundant in der Tabelle gespeichert. Die Redundanz wurde aus Performanzgründen eingeführt, um ein häufiges Parsen der XML-Strukturen zu umgehen. Die Maximalgrösse der XML-Struktur beträgt 16 Megabyte. Dies ist gemäss MySQL-Manual [28] die Grösse eines BLOB- resp. TEXT-Feldes.

Die Dokumenten-Historie wird in der Tabelle "DocHistory" verwaltet. Die Dokumenten-ID und das Einfügedatum dienen als Schlüssel. Das Einfügedatum ist als INT gespeichert und bestimmt die Anzahl Sekunden, die seit dem Jahr 2000 vergangen sind. Der Inhalt einer Datei wird als LONGBLOB ebenfalls in der Datenbank gespeichert und darf gemäss MySQL-Manual [28] 4 Gigabytes nicht überschreiten. Allerdings kann diese Kapazität bei weitem nicht ausgenutzt werden, denn das MySQL-interne Protokoll erlaubt eine maximale Paketgrösse von 16 MByte. Standardmässig läuft MySQL sogar mit einer Paketgrösse von nur 1 MByte. Damit grössere Pakete verwendet werden, musste der Parameter für die Paketgrösse geändert werden (siehe Abschnitt C.6). Somit können Dateien über 16 MByte nicht gespeichert werden. Mit MySQL 4.0 wird diese Schranke abgebaut und die maximale Paketgrösse auf 4 GByte erhöht (siehe MySQL-Manual [28]). (Zur Erinnerung: Eine weitere Schranke befindet sich auf Servlet-Seite im COS-Paket, welches einen HTTP-Header von maximal 1GB erlaubt.)

MySQL speichert jede einzelne ihrer Tabellen als Datei ab. Die Tabelle "DocHistory" kann aber eine gewaltige Grösse annehmen, wenn man bedenkt, dass hier alle Dokumente abgespeichert werden. Die Maximalgrösse einer MyISAM-Tabelle, welche standardmässig in MySQL zur Anwendung kommen, beträgt acht Millionen Terabytes ($=2^{63}$ Bytes). Eine weitere Hürde, die normalerweise früher greift, ist die des Betriebssystems. In der Windows-Welt gilt für FAT-Dateien eine Obergrenze von 2 GB, für FAT32-Dateien eine Obergrenze von 4GB und für NTFS-Dateien eine Obergrenze, die identisch mit der Festplattengrösse ist. In der Entwicklungsumgebung wurde Windows 2000 eingesetzt, welches als Dateisystem NTFS hatte. Wird in der produktiven Umgebung ein anderes Betriebssystem eingesetzt, muss auf dieses Limit geachtet werden. MySQL muss dann gegebenenfalls anders konfiguriert werden (siehe MySQL-Manual [28]). Ähnlich zur Dokumenten-Historie werden die zusätzlich hochgeladenen Dateien verwaltet. Als Schlüssel dienen die Dokumenten-ID und der Dateiname.

Um die Anbindung der Module zu demonstrieren, sind zwei Module entwickelt worden: Die Autoren-Notifikation und das Feedback-Formular. Beide besitzen eigene Tabellen in der Datenbank. Die Autoren-Notifikation speichert pro ID die Anzahl Tage, die ein Autor vor Ablauf des Dokumentes benachrichtigt werden

will. Dazu ist redundant das Datum eingetragen, zu dem der Autor benachrichtigt wird, um dem Prozess, welcher nach Notifikationszeitpunkte sucht, Rechenarbeit zu ersparen. Ändert ein Autor das Gültigkeitsdatum seines Dokumentes, so bleiben die Anzahl Tage fix und die Notifikationszeitpunkte werden neu berechnet. Bei der Notifikation wird immer das älteste Datum aus der Datenbank entfernt. Wird ein neues virtuelles Gegenstück erstellt, so werden automatisch die Werte ein Tag und sieben Tage eingetragen. Dadurch erhält der Autor sieben und einen Tag vor Ablauf seines Dokumentes eine Mitteilung, wann sein Dokument gelöscht wird. Anders verhält sich das Feedback-Formular. Das Erstellen neuer virtueller Gegenstücke wird vom Modul nicht behandelt. Wird aber ein Feedback-Formular für ein Dokument angefordert, das keine Einträge in den Feedback-Tabellen hat, so wird das Standardformular dynamisch erstellt.

Die Benutzerdaten werden in der Tabelle “user” gespeichert. Es werden die Benutzer-ID, das Passwort als Hashwert und die Email-Adresse abgelegt. Wie bei den Autoren ist die Gross-/Kleinschreibweise von ID und Passwort wichtig. Das Passwort wird mit der MySQL-eigenen Einwegfunktion “PASSWORD” gehasht und ist nicht rekonstruierbar. Besitzt ein Benutzer einen Barcode-Leser, der ihn identifiziert, wie z.B. bei AirClic über die Seriennummer, so hat jeder Barcode-Leser eine eigene Tabelle, mit deren Hilfe auf den Benutzer geschlossen werden kann. Im Falle von AirClic muss aus der Seriennummer auf die Benutzer-ID und das Passwort geschlossen werden. Aus diesem Grund speichert die Tabelle “AirClic” diese drei Werte. Das Passwort muss nochmals abgespeichert werden, weil es aus der “user”-Tabelle nicht rekonstruiert werden kann. Es darf nicht gehasht werden, sondern muss in Java ver- und entschlüsselt werden können. Die Benutzer-Historie wird in der Tabelle “History” gespeichert.

Die Tabelle “xsl” bestimmt für welchen ETHOC-Typ und welche Markup-Sprache welche XSLT-Datei(en) benutzt werden soll(en). In der Datenbank ist nicht der Dateiinhalt, sondern die URI der Datei ohne “.xsl”-Endung gespeichert. Der vollständige Dateinamen wird im Servlet (siehe Abschnitt 7.5 und 8.3.2), welches auf die Tabelle zugreift, zusammengestellt.

7.8 Aspekte der Verteiltheit

ETHOC wurde von Beginn an mit Blick auf eine spätere Verteiltheit konzipiert und umgesetzt. Alle Daten inkl. Dateien betreffend virtuelle Gegenstücke werden in der Datenbank gespeichert. Somit sind die Daten losgelöst von der Anzahl der Installationen und dem Ort der ETHOC-Server. Die URI der Datenbank wird einem ETHOC-Server über eine Konfigurationsdatei bekanntgemacht. D.h. die ETHOC-Java-Pakete können auf mehrere Rechner installiert werden und alle würden auf die gleiche Datenbank zugreifen. Kritische Operationen, wie das Vergeben der Koordinatoren-IDs, operieren mit Datenbank-Locks auf den entsprechende Tabellen. Ebenfalls in einer Konfigurationsdatei befindet sich der Namen des SMTP-Servers. Bei einer späteren Verteiltheit der ETHOC-Server muss bedacht werden, dass diverse Dispatcher für die Zugriffe der ETHOC-Browser benötigt werden. Der ETHOC-Browser adressiert gewisse Servlets, die er benötigt, direkt. Welche Servlets betroffen sind, ist im Javadoc der Klasse “Ethoc.System.Constants” beschrieben.

Die Datenbank selbst könnte später ebenfalls verteilt organisiert werden. Diese Änderung betrifft aber die Applikations-Seite nicht (oder nur den Datanbank-Manager). Auf die Datenbank greift die Applikation immer über eine Klasse, den Datenbank-Manager, zu (Erklärung siehe Abschnitt 6.5.1). Dieser ist aus Performanzgründen als Singleton implementiert. Der Aufbau einer Verbindung zur Datenbank ist sehr zeitintensiv. Wird immer der gleiche Datenbank-Manager benutzt, was bei einem Singleton der Fall ist, entfällt diese Wartezeit. Da der Datenbank-Manager nur vom Server aus benutzt wird, spielt die Bandbreite der Internetanbindung der Klienten keine Rolle. Die Tatsache, dass der Datenbank-Manager als Singleton implementiert worden ist, impliziert keine Einschränkung auf eine spätere Verteiltheit, denn er ist pro ETHOC-Server einmalig.

7.9 Sicherheitsaspekte

Ziel dieser Arbeit ist, einen Prototyp zu erstellen, der die gewünschte Funktionalität sicherstellt und die Benutzbarkeit des Systems demonstriert. Sicherheitsaspekte spielen eine sekundäre Rolle. Dennoch wurde darauf geachtet, grosse Sicherheitslücken zu vermeiden.

7.9.1 Ausgezeichnete Benutzerrechte

Alle Daten befinden sich in der Datenbank. Der Zugriff auf die Datenbank ist auf drei Benutzer reduziert worden. Dies sind “root”, welcher alle Rechte hat, “ethocTRX”, welcher schreibend auf die ETHOC-Datenbank zugreifen darf und “ethocSEL”, welcher nur ETHOC-Daten lesen darf. Alle anderen in MySQL standardmässig definierten Benutzer sind entfernt worden.

Der Benutzer “root” wird vom Datanbankadministator benutzt und wird im Normalbetrieb nicht benötig. Der technischen Benutzer “ethocTRX” wird vom Datanbank-Manager benötigt (siehe Abschnitt 6.5.1). “EthocSEL” wird im Moment nicht benutzt.

7.9.2 Datenübertragung

Im Prototypen werden alle Daten unverschlüsselt übers Internet zum Web-Server und zur Datenbank übertragen. Im Benutzer-Portal werden im Falle eines HTML-Browsers Benutzer-ID und Passwort in Cookies, im Falle eines WML-Browsers als WML-Variablen gespeichert. Letztere haben aber einen globalen Sichtbarkeitsbereich, d.h. sie sind über Domain-Grenzen hinweg auslesbar.

Die Navigation innerhalb der Benutzer-Historie geschieht in WML mittels GET-Anfragen, indem auf einen Link geklickt wird. Dabei wird Benutzer-ID und Passwort im Klartext innerhalb der URL übergeben. Der Grund liegt darin, dass diese Art der Navigation benutzerfreundlicher ist, als die Angaben über eine POST-Anfrage zu senden. Bei POST-Anfragen geschieht die Navigation über das Menü oder der Options-Taste und dem Suchen nach dem gewünschten Befehl.

Der ETHOC-Browser überträgt ebenfalls Benutzer-ID und Passwort zum Server. Dies bei allen Operationen, welche die Synchronisation der Online-Historie

und der lokalen Historie betreffen. Momentan werden für die Kommunikation zwischen ETHOC-Browser und ETHOC-Server GET-Requests verwendet und die Parameter in der URL kodiert. Ursprünglich wurden POST-Requests verwendet. Doch bei vielen schnellen Request hintereinander tauchte ein Fehlverhalten auf. Nach etwa vier Anfragen blockierte der Server und nahm keine Anfragen mehr an. POST-Requests funktionierten nur dann einwandfrei, wenn kleine Pausen dazwischen geschaltet wurden. Gemäss den Bug-Reports von SUN bei ähnlichen Beobachtungen konnte das Fehlverhalten auf die Web-Server zurückgeführt werden. Es könnte sich hier also um einen Bug in Jakarta handeln. Nach der Umstellung auf GET-Requests funktionierte alles einwandfrei. Dadurch entstand aber das erwähnte Sicherheitsproblem.

7.9.3 Passwörter

Die Passwörter der Autoren und der Benutzer des ETHOC-Systems werden als Hashwert (siehe Abschnitt 7.7) in der Datenbank abgelegt, so dass es nicht einmal dem Datenbank-Administrator möglich ist, Passwörter zu rekonstruieren. Die Passwörter dienen zur Identifikation und nicht als Zugriffmechanismus auf die Datenbank. ETHOC-Programme greifen mit einer technischen Benutzer-ID auf die Datenbank zu (siehe Ausführungen in Abschnitt 6.5.1), deren Passwort nur dem Datenbank-Manager bekannt ist. Der Grund, weshalb die Passworte nicht in der Klasse “Constants” sind, liegt darin, dass sie dort nicht privat sondern protected wären. Da das System-Paket auch Bestandteil des ETHOC-Browsers ist, der lokal installiert ist, könnte ein versierter Benutzer das Passwort herausfinden, um auf die Datenbank schreibend zuzugreifen, und so beliebige Daten verändern.

Wird ein AircClic-Leser verwendet, so wird das Benutzer-Passwort nochmals abgespeichert. Diesmal geschieht die Verschlüsselung in Java. Das Passwort muss in Java auch entschlüsselt werden können. Nur dadurch kann der Benutzer automatisch eingeloggt werden. Im Prototyp ist das Passwort im Klartext abgelegt. Wünschenswert wäre, wenn die Benutzerdaten auf eine andere Weise organisiert werden könnten, zum Beispiel, indem sie vom Leser verschlüsselt mitgeliefert würden. Doch AirClic überträgt lediglich die Seriennummer des Gerätes und den eingelesenen Barcode. Dies geschieht sogar mittels einer GET-Anfrage, d.h. die Parameter werden in der URL übertragen. Somit ist es ein leichtes Spiel, die Seriennummer abzuhören oder zu erraten und sich unerlabterweise als “böswilliger” Benutzer mit dieser Seriennummer manuell anzumelden.
Sobald ETHOC über einen Prototypen hinausgeht, muss sich überlegt werden, ob Anbindungen, von Barcode-Lesern wie AirClic, die zu wenig Informationen übertragen, auf diese Art unterstützt werden soll. Im Moment würden sich zwei Lösungsansätze anbieten. Man verzichtet auf die implizite Authentifizierung durch Serienummern, oder man versucht auf persönliche Daten auf dem Mobiltelefon, wie SIM-Card oder bestimmte Einträge im Telefonbuch (die z.B. die Zugangsdaten erhalten) zuzugreifen und dadurch die Authentifizierung zu ermöglichen. Beides scheint zum heutigen Zeitpunkt aufgrund fehlender Schnittstellen nur eingeschränkt realisierbar.

Im lokalen ETHOC-Browser wird das Objekt “UserData”, welches die Präferenzen des Benutzers und seine Daten enthält, serialisiert, sobald der Browser

geschlossen wird. Beim nächsten Start wird das serialisierte Objekt wieder in den Browser eingebunden. Somit wird der Browser beim Neustart so eingerichtet, wie er zuvor verlassen wurde (siehe Abschnitt 7.6). Eine Variable in "UserData" ist das Passwort des Benutzers. Diese wird verwendet, um den Benutzer zu verifizieren, falls eine Online-Verbindung fehlt und um ihn automatisch gegenüber dem ETHOC-Server auszuweisen, wenn die Online-Historie verändert wird. Das Passwort wird also mitserialisiert. Hat jemand Zugriff auf die Datei, kann er das Passwort mit geeigneten Werkzeugen extrahieren. Dieses Sicherheitsloch kann später wie folgt behoben werden: Vor dem Serialisieren wird ein Hashwert des Passwortes gespeichert und das richtige Passwort gelöscht. Bei erfolgreichem Login kann dann das Passwort wieder eingetragen werden.

Kapitel 8

Erweiterung des ETHOC-Systems

Das ETHOC-System kann auf mehrere Arten erweitert werden. Zum Beispiel könnte mehr Funktionalität für virtuelle Dokumente implementiert werden. Abschnitt 8.1 erklärt, wie neue Module, die diese Funktionalität bieten, in ETHOC integriert werden können. Ein anderer Ansatz ist, das Prinzip auf Gegenstände auszuweiten, also ETHOC konzeptionell zu erweitern. Abschnitt 8.2 diskutiert, inwieweit die vorhandene Implementierung dies ermöglicht. Abschnitt 8.3 erklärt, wie allfällige neue Typen von virtuellen Gegenstücken in ETHOC integriert werden können. Im Abschnitt 8.4 wird gezeigt, wie neue Typen von Barcode-Leser eingebunden werden können.

8.1 Neue Module einbinden

ETHOC-Dokumente mit neuen Funktionen für Autor und Benutzer erweitern, geschieht am einfachsten über die Implementierung neuer Module. Hierfür muss der ETHOC-Kode nicht angepasst werden. Ein Modul-Entwickler muss eine Internet-Seite programmieren, die der Autor eines virtuellen Gegenstückes benutzt, um das Modul für sein Dokument zu konfigurieren. Diese muss nicht gezwungenermassen über Java-Servlets geschehen, erleichtert aber die Arbeit, da die Servlets von "Ethoc.Authoring.EthocServlet" abgeleitet werden können. Die ganze Infrastruktur zum Speichern der modulbezogenen Daten muss ebenfalls entwickelt werden. Dabei gilt es zu beachten, dass ein Autor nicht gezwungen werden kann, ein Modul zu konfigurieren. Er kann es einfach nur aktivieren und nichts eingeben. Der Entwickler eines Moduls muss sicherstellen, dass sein Modul für diesen Fall ebenfalls eine sinnvolle Handlung durchführt resp. den Benutzern eine sinnvolle Ausgabe anzeigt, sofern das Modul auch einen Einstiegspunkt für Benutzer besitzt.

Beim Einstiegspunkt für die Benutzer muss daran gedacht werden, dass nicht nur HTML-Browser, sondern auch WML-Browser Kundenapplikationen sind und nach Möglichkeit beide unterstützt werden sollen. Um den Benutzer zu einem

HTML- resp. WML-Servlet umzuleiten, kann die Klasse “Ethoc.System.AgentChecker” verwendet werden. Diese Analysiert den User-Agent und gibt zurück, ob der Browser HTML oder WML versteht.

Soll ein Modul informiert werden, wenn es ein- oder ausgeschaltet wird, oder wenn sich ein virtuelles Gegenstück ändert, so muss eine Klasse geschrieben werden, die das Interface “Ethoc.System.ModulNotifier” implementiert. Alternativ kann die Klasse “Ethoc.System.ModulNotifierImpl” erweitert werden und dort nur die Methoden überschrieben werden, die auch benötigt werden. Ein ModulNotifier wird beim Start des Systems instanziert und vom System verwaltet. Ein allfälliges Starten von benötigten Hintergrundsprozessen soll deshalb im Konstruktor des Notifiers erfolgen.

Das neue Modul und sein Notifier werden dem System durch einen Eintrag in der Datei “modules.xml” bekantgemacht (siehe Abschnitt 7.1.2). Es darf nicht vergessen werden, die Versionsnummer des Wurzel-Elementes hochzuzählen, sonst werden die neuen Module beim Ändern von bestehenden virtuellen Gegenstücken nicht eingebunden. Es wird nämlich aus Performance-Gründen nur das Attribut “last_config” verglichen (siehe Abschnitt 6.4.4).

8.2 Erweiterung auf Gegenstände

Die virtuellen Gegenstücke der ETHOC-Dokumente haben den Typ “doc”, und in diesem Bericht ist des öffteren von virtuellen Dokumenten die Rede. Doch die in dieser Arbeit erstellten virtuellen Gegenstücke lassen sich auch für andere physische Objekte verwenden. Dabei erweist sich die gewählte und implementierte Variante des Autorenportals als Vorteil. Es werden Barcodes als Bild zurückgegeben, und somit sind die ETHOC-IDs losgelöst von physischen Dokument. Die Barcodes können direkt auf Etiketten gedruckt und an Objekte angebracht werden. Dabei gilt es aber das Problem zu lösen, dass Vandalen die Barcodes entfernen oder eigene anbringen können.

Ein Sitzungsraum, der mit einer ETHOC-ID verknüpft ist, könnte in seinem virtuellen Gegenstück einen Link auf das Reservationssystem anbieten. Somit müsste ein Benutzer nichts anderes tun als den Barcode einzuscannen und auf den Link zu klicken. Alternativ könnte ein Reservationsmodul implementiert werden. Von diesem Modul könnten z.B. neben der Raumbewirtschaftung auch die Bibliotheken Gebrauch machen.

Ein virtuelles Gegenstück eines Kopierers könnte zum Beispiel für die Benutzer Links zu einer kurzen Bedienungsanleitung oder zu einem interaktiven Führer anbieten. Für den Service-Monteur könnte ein Link zur passwortgeschützten Revisionsseite und zum Ersatzteillager angebracht sein.

Doch dem “doc”-Typ sind Grenzen bei der Modellierung von Objekten gesetzt. Dies soll am Beispiel eines Druckers mit virtuellem Gegenstück erläutert werden. An einer Hochschule wie der ETH stehen zahlreiche Drucker, verteilt über viele Gebäude. Angenommen ein Student befindet sich vor einem Drucker, den er nie zuvor benutzt hat und möchte ein Dokument drucken. Das virtuelles Gegenstück könnte die Druckparameter mitteilen oder den Druckertreiber zum Download anbieten. Dies ist problemlos mit dem “doc”-Typ möglich. An der ETH können Dateien über eine Web-Schnittstelle an jeden öffentlichen Drucker

gesendet werden. (siehe http://vpp-router1.ethz.ch/vpp.html). Das virtuelle Gegenstück könnte dann einen Link auf diese Seite anbieten und seine Parameter gleich selbst eintragen. Auch dies wäre mit dem "doc"-Typ möglich. Folgendes Szenario ist aber nicht möglich: Angenommen der Student hat zuvor einen ETHOC-Barcode von einem Dokument eingescannt und möchte dieses auf dem Drucker ausdrucken. Der aufmerksame Leser würde jetzt fragen, wo das Problem liege, man könne doch die Online-Kopie runterladen und dann das Dokument mit den oben beschriebenen Mechanismen ausdrucken. Dies ist richtig, doch was passiert, wenn der Barcode-Leser nicht mit einem PC, sondern einem Mobiltelefon kommuniziert? Auf einem Mobiltelefon können keine Dateien heruntergeladen werden (oder zumindest noch nicht). Das virtuelle Gegenstück des Druckers müsste also auf die Historie des Benutzers zurückgreifen und fragen, welche Dateien gedruckt werden sollen. (Eine Variante wäre, dass bei einem zweifachen Einscannen des Druckerbarcodes automatisch das letzte Dokument aus der Historie gedruckt würde.) Der "doc"-Typ kann dies aber nicht leisten. Für dieses Szenario müsste ein neuer ETHOC-Typ entwickelt werden, genauer ein Typ der mit der Historie des Benutzers arbeitet.

Es mag auf den ersten Blick etwas aufwendig klingen, für diesen kleinen Sonderfall gleich einen neuen Typ zu definieren. Doch vielleicht wäre ETHOC mit dem Drucker-Typ zusammen sogar die Killerapplikation im Bereich des Verknüpfens von Alltagsgegenstände mit dem Internet. Man stelle sich vor, es stehen zwei Personen im Gang und diskutieren über ein Paper der ersten Person. Die zweite Person möchte gerne eine Kopie davon. Also scannt sie mit ihrem Mobiltelefon die ETHOC-ID des Papers geht zum Drucker und scannt dessen ID zweimal ein, und schon ist das Dokument gedruckt.

Zusammengefasst kann gesagt werden, dass mit dem "Doc"-Typ auch Objekte sehr gut modelliert werden können. Doch zusätzliche Typen, wie der obige Drucker-Typ, würden weiterführende Szenarien erlauben. Der folgende Abschnitt erklärt nun, wie neue ETHOC-Typen ins System eingebaut werden können.

8.3 Neue ETHOC-Typen

8.3.1 Autorenseitig

Wie in Abschnitt 7.1.1 erklärt, beginnt die XML-Definition des virtuelle Gegenstück mit dem Wurzelement "ethoc", welches ein Attribut "type" besitzt. Es muss nun eine neue Struktur erstellt werden, die "ethoc" als Wurzelement und einen neuen noch nicht verwendeten Wert im Attribut "type" besitzt. Dazu müssen passende Werkzeuge entwickelt werden, um die virtuellen Gegenstücke des neuen Typs zu erstellen resp. zu ändern. Dem Entwickler neuer Werkzeuge steht frei, neue Java-Pakete oder sogar eine andere Programmiersprache zu verwenden. Die neuen Werkzeuge werden dem System bekannt gemacht, indem ihre Einstiegspunkte und Namen in der Datei "authoringtools.xml" notiert werden Alle ETHOC-Typen werden in der Datenbank in der Tabelle "doc" gespeichert, in welcher nebst der XML-Struktur noch redundante Informationen abgelegt werden (siehe Abschnitt 7.7).

8.3.2 Benutzerseitig

Greifen Benutzer über das Benutzer-Portal zu und fügen dort neue ETHOC-IDs in ihre Historie ein, so muss das Verhalten beim Auftreten von Duplikate modelliert werden. Zum Beispiel macht es für IDs vom Typ "doc" keinen Sinn, mehrfach in die Historie aufgenommen zu werden, für den oben skizzierten Drucker-Typ aber vielleicht schon. Zweimal die selbe Drucker-ID hintereinander könnte bedeuten, dass der Benutzer das letzte Dokument automatisch drucken lassen will, während ein einmaliges Einlesen der Drucker-ID, das Drucken eines auszuwählenden Dokumentes innerhalb der Historie symbolisieren mag. Das gewünschte Verhalten beim Einfügen in die Benutzer-Historie wird durch Klassen modelliert, die die abstrakte Klasse "HistoryAdder" erweitern. Anhand des ETHOC-Typs ruft das System den zugehörigen "HistoryAdder" auf. Dieser wird über Namenskonvention bestimmt. Der Name besteht darin, dass der Typ in Grossbuchstaben an "HistoryAdder" angefügt wird. Wird also ein neuer ETHOC-Typ namens "xyz" eingeführt, so muss es eine Klasse "HistoryAdderXYZ" mit den oben beschriebenen Eigenschaften implementiert werden.

Das Resultat der ID-Auflösung ist ein XML-, WML- oder HTML-String. Dies geschieht durch eine XSL-Transformation. Für jeden ETHOC-Typ und jede Markup-Sprache müssen ein oder mehrere XSLT-Dokumente erstellt werden. In der Datenbank wird die URI der Dateien (ohne Endung ".xsl") eingetragen. Gibt es für eine Markup-Sprache mehrere XSLT-Dateien (wie z.B. WML im Typ "doc"), die je einen speziellen Teil anzeigen und zwischen deren Ausgaben navigiert werden kann, sollten sie einen gemeinsamen Stamm im Dateinamen haben. Als URI in die Datenbank wird der Pfad plus der Stamm eingetragen. Erhält das Servlet "ShowID" den Parameter "part", so wird an die URI der Parameter und die Endung ".xsl" konkateniert und so der Dateiname konstruiert. Existiert kein Parameter namens "part", so wird lediglich die XSL-Endung angefügt.

Für den ETHOC-Browser müssen Model- und View-Objekt geschrieben werden. Die neuen Klassen werden über Namenskonvention vom Browser gefunden. Heisst der neue ETHOC-Typ "xyz", so müssen die Klassen "ModelXYZ" und "ViewXYZ" benannt werden.

8.4 Einbinden neuer Barcode-Leser

Wird der Barcode-Leser auf einem System betrieben, auf dem der ETHOC-Browser nicht installiert werden kann, z.B. weil keine Java-Edition für das System existiert, so muss der Barcode-Leser den HTML- oder WML-Browser öffnen und den eingelesenen Barcode mit weiteren Parametern einem Servlet übergeben. Es gibt also eine serverseitige Anbindung und eine Anbindung im ETHOC-Browser.

8.4.1 Serverseitige Anbindung

Am einfachsten ist es, den HTML-Browser des Benutzers zu öffnen, und das Servlet "Ethoc.Resolving.LoginHTML" zu adressieren. Dabei muss der Parame-

ter “next” den Wert “Ethoc.Resolving.AddIDHTML” und “docID” den Wert der ETHOC-ID enthalten. Weitere optionale Parameter sind “anonymus”, welcher dem Benutzer erlaubt auf einen Login zu verzichten, und “intro”, welcher eine Erklärung einblendet, dass ein Login zur Verwaltung der persönlichen Online-Historie erforderlich ist. Analog kann ein WML-Browser benutzt werden. Für HTML-Browser besteht alternativ die Möglichkeit, das Servlet “Ethoc.Resolving.AddIDHTML” mit den Parametern “docID” (der ETHOC-ID), “id” (der Benutzer-ID) und “pass” (dem Benutzer-Passwort) zu adressieren. Wird der Parameter “cookies” mit dem Wert “on” übergeben und der Browser unterstützt Cookies, so werden die Benutzerdaten in Cookies gespeichert und der Benutzer muss sich für weitere Zugriffe nicht mehr ausweisen.

Ein aufwendigerer Weg besteht darin ein eigenes Servlet zu programmieren, das Vorarbeit leistet. Zum Beispiel indem es den Benutzer automatisch authentisiert, die Angaben in Cookies oder WML-Variablen speichert und dann erst zum Login-Servlet umleitet. Wird eine indirekte Authentisierung gewählt, wie z.B. in AirClic über die Seriennummer (siehe Abschnitt 7.5), muss ein Matching zwischen propriäterer Authentisierung und ETHOC-Benutzer-ID mit Passwort umgesetzt werden. Hierfür muss eine Tabelle in der Datenbank erstellt werden, welche das Attribut “user” mit dem Inhalt die ETHOC-Benutzer-IDs enthält. Der Name der neuen Tabelle muss in der Konfigurationsdatei “BCReader.cfg” auf dem Server eingetragen werden. Dadurch kann der Benutzer beim Bearbeiten seines Profils Einträge von Barcode-Leser, die ihn indirekt authentisieren, löschen, z.B. wenn die entsprechenden Barcode-Leser sich nicht mehr in seinem Besitz befinden.

8.4.2 Anbindung im ETHOC-Browser

Wird der Barcode-Leser über den ETHOC-Browser verwendet, so muss lediglich eine Klasse geschrieben werden, die das Interface “Ethoc.Browser.IDReader” implementiert. Der Benutzer kann den Barcode-Leser verwenden, wenn er im Konfigurationsfenster den Namen der neuen Klasse (inkl. Paket) eingibt.

Kapitel 9

Diskussion und Ausblick

Dieses Kapitel fasst die Arbeit zusammen. Die Arbeit wird bewertet und als Ausblick weitere Arbeiten vorgeschlagen.

9.1 Erfüllung der Aufgabenstellung

Die Ziele der Diplomarbeit (siehe Anhang A) sind erreicht worden.

1. Autorenseite: Für die Autoren wurde als webbasiertes Werkzeug das Autorenportal erstellt. Der Druckertreiber wurde nach der Analysephase wegen des immensen erwarteten Implementierungsaufwandes auf eine spätere Arbeit verschoben.

2. XML-Schema: Die virtuellen Gegenstücke werden in XML-Form verwaltet (siehe Anhang D.1).

3. Zuordnung Dokument und Funktionen: Mögliche Online-Funktionalitäten werden innerhalb der XML-Struktur verwaltet (siehe Abschnitt 5.5.1). Um neue Funktionalitäten, die es zum heutigen Zeitpunkt nicht gibt, zu ermöglichen, wurde die Idee von Modulen umgesetzt (siehe Abschnitte 5.5.2, 7.3 und 8.1).

4. Verwaltung: Zur Verwaltung der Online-Ressourcen zu gedruckten Dokumente dient die Datenbank.

5. Mapping-Service: Das Mapping von ID zur XML-Struktur geschieht in einem Servlet, das die Datenbank zu Hilfe nimmt.

6. Benutzungskonzepte: In Abschnitt 2.1 sind motivierende Szenarien vorzufinden. Zur Verifikation der Implementierung sind Szenarien in Abschnitt 4.1 aufgestellt worden, die anschliessend in Abschnitt 9.3 diskutiert werden. Sie sind mit dem ETHOC-System alle realisierbar.

7. Skizzierung einer Erweiterung: Die vorliegende Implementierung eignet sich nicht nur für gedruckte Dokumente, sondern auch für andere physische Objekte (siehe Abschnitt 8.2). Genügen die bereitgestellten Mechanismen

den Anforderungen nicht, kann ETHOC um neue Typen erweitert werden (siehe Abschnitt 8.3).

9.2 Bewertung der Arbeit

Mit dieser Arbeit ist als Grundgerüst ein System entstanden, welches das Verwalten und Vermitteln von beliebigen virtuellen Gegenstücken ermöglicht. Als eine Ausprägung von virtuellen Gegenstücken wurde der Dokument-Typ implementiert. Der Dokument-Typ selbst besitzt als weitere Flexibilität die Anbindung von Modulen. Module können aber bei Bedarf auch für andere Typen eingesetzt werden. Das Einbinden neuer Typen oder Module geschieht über Namenskonventionen und Konfigurationsdateien (siehe Kapitel 8).

Das System ist plattformunabhängig, da die Java-Technolgie verwendet worden ist. Klienten, für die keine Java-Edition existiert, können über eine WML- oder HTML-Schnittstelle auf die virtuellen Gegenstücke zugreifen. Ein Benutzer hat eine globale Historie. IDs, die er z.B. mit dem Mobiltelefon eingelesen hat, stehen ihm später auch im ETHOC-Browser oder im HTML-Portal zur Verfügung.

Der lokale ETHOC-Browser besitzt einen Cache, in welchen die Dokumente und Dateien heruntergeladen werden. Der Cache ist für alle Benutzer eines Rechners gemeinsam. D.h. wenn jemand eine Datei heruntergeladen hat, steht sie allen zur Verfügung. Jeder Benutzer des ETHOC-Browsers kann eigene Kommentare an ein virtuelles Gegenstück anbringen, die vom ETHOC-Browser verwaltet werden und die für die anderen Benutzer natürlich nicht sichtbar sind.

Zu Beginn der Arbeit, war ein Schlagwort “automatische Aktionen”. Der Benutzer liest einen Barcode ein und es geschieht etwas automatisch auf seinem Gerät. Dies funktioniert nicht für HTML- und WML-Browser, da diese keine automatischen Aktionen erlauben (siehe Abschnitt 5.4). Einzig für den ETHOC-Browser wären automatische Aktionen möglich. Statt Dateien automatisch zu öffnen und Kalendereinträge einzutragen, wurde als einziger Automatismus, nebst der XML-Verwaltung der IDs, ein Dateien-Cache eingerichtet. Dateien, die eine vom Benutzer festgelegte Grösse nicht übersteigen, werden automatisch heruntergeladen. Erst auf Verlangen des Benutzers werden sie geöffnet oder in ein gewünschtes Zielverzeichnis kopiert. Kalendereinträge sind Bestandteil der XML-Struktur. Auch sie werden erst auf Verlangen hin extrahiert und eingetragen resp. gespeichert.

9.3 Verifikation der Benutzungsszenarien

In Abschnitt 4.1 sind Benutzungsszenarien aufgestellt worden, mit dem Zweck die Arbeit später an diesen zu verifizieren – was der Inhalt dieses Abschnittes ist.

9.3.1 Filmveranstaltung und drahtlose Kommunikation

Dieses Szenario (siehe Abschnitt 4.1.1) kann wie folgt mit ETHOC umgesetzt werden: Der Student hat auf seinem Laptop den ETHOC-Browser und den

Treiber für seinen Barcode-Leser installiert. Der Aufruf zwischen Treiber und ETHOC-Browser wurde bereits im Design anders konzipiert, als es im Szenario vorgestellt wurde. Es ist nicht der Treiber, der den ETHOC-Browser aufruft, sondern umgekehrt. Dadruch werden die IDs nur dann übertragen, wenn der ETHOC-Browser bereits gestartet ist. Diese Art der Kommunikation erlaubt dem Benutzer seinen Barcode-Leser mit anderen Applikationen zu benutzen, falls er andere Barcodes einliesst.

9.3.2 Vorlesung und Studenten-Raum

Auch dieses Szenario (siehe Abschnitt 4.1.2) ist in ETHOC möglich: In diesem Fall ist der ETHOC-Browser auf den PC installiert und der Student hat den richtigen Treiber installiert. Der Aufruf zwischen Barcode-Leser und ETHOC-Browser ist, wie im Abschnitt zuvor erwähnt, in der anderen Richtung implementiert worden. Im Gegensatz zum ursprünglichen Szenario besitzt der ETHOC-Browser einen Cache. Die Dateien werden also nicht in ein vom Benutzer gewähltes Verzeichnis kopiert, sondern in den Cache. Der Benutzer kann dann die Dateien direkt öffnen oder sie zur Bearbeitung in ein Verzeichnis seiner Wahl kopieren lassen.

Änderungen an einem virtuellen Gegenstück werden einem Benutzer durch die Meldung “update” in der ID-Übersicht im oberen Breich des ETHOC-Browsers angezeigt. Im Falle von Dokumenten, Dateien, Kalendereinträgen, Modulen und Links wird bei einer Änderung angezeigt, welches Element sich verändert hat, oder hinzugekommen ist. Somit ist ein Benutzer leicht über Änderungen informiert. Dateien, die nicht veröffentlich sind (wie in diesem Szenario die Musterlösung), befinden sich weder im Cache, noch kann eine Interaktion mit ihnen gestartet werden. Erst ab dem vom Autor festgelegten Publikationszeitpunkt werden sie, sofern sie die vom Benutzer festgelegte Maximalgrösse nicht überschreiteen, heruntergeladen.

9.3.3 Aktionen auf Kleinstrechner

Dieses Szenario (siehe Abschnitt 4.1.3) wurde ebenfalls realisiert: Es wurde mit einem Rechner gearbeitet, auf dem der ETHOC-Browser nicht installiert werden kann, weil keine Java-Edition verfügbar ist. Allerdings existiert ein HTML-Browser. Somit startet der Barcode-Leser den Browser mit der URL des Benutzer-Portals gemäss Abschnitt 8.4.1. Da es sich, um einen HTML-Browser handelt, werden die PDF-Dateien zum Downlaod angeboten, unabhängig davon, ob der Rechner die Datei öffnen kann oder nicht. Die Unterscheidung der Klienten wurde also grob, aber für die Praxis einsatztauglich, umgesetzt. Wie in Abschnitt 7.5 besprochen, werden die Klienten nach den Sprachen XML, HTML und WML kategorisiert. Nur den WML-Klienten steht der Download von Dateien generell nicht zur Verfügung.

9.3.4 Annonce und Mobiltelefon

Das Szenario ist wie in Abschnitt 4.1.4 beschrieben mit ETHOC realisierbar.

9.4 Vorschläge für weiterführende Arbeiten

Mit dieser Arbeit ist ETHOC nicht abgeschlossen. Es sind einige interessante Arbeiten möglich, die auf diese Arbeit aufbauen.

9.4.1 Modifizierende Arbeiten

Diese Arbeiten umfassen das Studium und das Abändern von bestehendem Kode.

- Die Kalendereinträge werden im Moment über Eingabefelder vom Autor abgefragt und dann in die XML-Struktur eingebaut. Ein anderer Ansatz wäre, dem Autor zusätzlich zu ermöglichen, vCalendar-Dateien (*.vcs) hochzuladen. Diese würden durch das System geparst und dann in die XML-Struktur integriert.
- Als Online-Kopien der Dateien werden im Moment alle Formate unterstützt. Im Hinblick auf spätere Drucker-Objekte wäre eine Einschränkung auf PDF und PostScript denkbar. Hierfür müsste dem Autor ein Werkzeug zur Verfügung gestellt werden, mit welchem seine Dokumente automatisch konvertiert werden.
- Der logische Online-Drucker (siehe Abschnitt 5.3.3) könnte in dasjenige Servlet eingebaut werden, das den Barcode anzeigt. Somit könnte ein Autor wählen, ob er den Barcode selbst in seine Applikation einfügen will, oder ob das System dies für ihn machen soll.
- Unterstützung standardisierter Barcodes, wie EAN13.

9.4.2 Erweiternde Arbeiten

Diese Arbeiten setzen die Kenntnis der Namenskonventionen und der Struktur der Konfigurationsdateien voraus. Der ETHOC-Kode wird dabei nicht verändert.

- Der lokale logische Drucker (siehe Abschnitt 5.3.2) könnte als zusätzliches lokales Werkzeug implementiert werden. Dabei würden nur die minimal notwendigen Eingaben vom Autor erfragt und dieser dann aufs Autoren-Portal verwiesen werden.
- Entwicklung weiterer Typen, wie der in Abschnitt 8.2 erwähnte Drucker-Typ, einen Link-Typ, welcher einen Barcode mit einer Internet-Seite verbindet, oder einen Visitenkarten-Typ, der die Person ins Adress- oder Telefonbuch einträgt.
- Entwicklung weiterer Module für den Dokumenten-Typ, wie z.B. ein Modul, das die Anzeige eines bestimmten Dokumentes auf einen zu bestimmenden Benutzerkreis limitiert oder ein Modul, das die Anmeldung an Veranstaltungen ermöglicht.
- Verteilte Organisation der Datenbank umsetzen.

Danksagung

Am Ende der Diplomarbeit und somit meines Studiums angelangt, widme ich diese Arbeit meiner Mutter und meinen Grosseltern, die mir das Studium ermöglicht haben.

Ich bedanke mich bei meinen Betreuern Jürgen Bohn und Michael Rohs für die fruchtbaren inhaltlichen Diskussionen, das Zurverfügungstellen der für diese Arbeit benötigten Infrastruktur und für das Korrekturlesen dieses Berichtes.

Herzlichen Dank an alle.

Anhang A

Aufgabenstellung

Thema dieser Diplomarbeit ist die Erweiterung von gedruckten Dokumenten – wie z. B. Postern, Veranstaltungshinweisen, Zeitschriftenartikeln oder Lehrmaterialien – um Online-Informationen und Online-Funktionalität. Als Informationsanker vom physischen Dokument zur virtuellen Erweiterung sollen Barcodes verwendet werden. Diese haben den Vorteil, dass sie sich mit herkömmlicher Technik einfach ausdrucken und kopieren lassen und weder spezielle Hardware, noch spezielle Printmedien erfordern. Die Kosten für diese Art von "physischen Hyperlinks" sind also sehr gering. Wesentliche Bestandteile der Arbeit sind Konzepte für das Auflösen der gelesenen Identifikationsnummer, das Anzeigen und die Interaktion mit den zugeordneten Daten und Operationen, die Verknüpfung von Online-Inhalten und deren Verwaltung und die Ausarbeitung von einigen exemplarischen Benutzungsszenarien. Durch die zu entwickelnde Infrastruktur soll einerseits der Autor beim Erstellen eines Dokumentes und der zugehörigen Funktionalität, andererseits der Leser, der ausgehend vom ausgedruckten Dokument auf dessen Online-Repräsentation oder weitere zugeordnete Ressourcen zugreifen will, unterstützt werden.

Szenarien

Ein Anwendungsszenario könnte wie folgt aussehen: Aushänge, Plakate und Flyer werden mit Barcodes versehen. Diese lassen sich mit einem mobilen Gerät, wie einem PDA, einem Mobiltelefon oder einem am Schlüsselbund befestigten – eventuell drahtlos kommunizierenden – Barcodescanner lesen. Das Übermitteln der gelesenen Identifikationsnummer löst dann bestimmte Aktionen in der zu entwickelnden Infrastruktur und auf dem Gerät des Benutzers aus. So könnten Reservierungen vorgenommen, Kalendereinträge erstellt und Detailinformationen abgerufen werden. Hierbei bestehen kaum Beschränkungen bezüglich der Art der gelieferten Informationen oder der ausgelösten Aktionen. Sowohl textuelle, als auch auditive und grafische Ausgaben sind denkbar.

Ein zweites Szenario ist durch die Einbettung von Barcodes in Zeitschriftenartikeln oder wissenschaftlichen Publikationen gegeben. Der Barcode könnte zu einem vom System verwalteten "Portal" für den Artikel führen, auf dem eine

elektronische Version verfügbar ist, auf dem über den Artikel diskutiert wird oder auf dem auf thematisch verwandten Arbeiten verwiesen wird. Hierbei sollte der Autor steuern können, welche Aktionen mit seinem Dokument verknüpft sind, bzw. welche weiteren Aktionen und Inhalte Leser hinzufügen dürfen.

Der Lehrbetrieb der ETH bietet weitere Anwendungsmöglichkeiten, z.B. das Anknüpfen von Online-Inhalten an Vorlesungs-, Übungs- und Praktikumsunterlagen. So könnte ein auf dem Übungsblatt aufgedruckter Barcode den Zugriff auf Beispiellösungen ermöglichen. Die Infrastruktur würde dabei sicherstellen, dass die Lösungen erst ab einem bestimmten Zeitpunkt freigegeben werden. In Vorlesungsunterlagen liessen sich multimediale Lehrinhalte einbauen, die durch das Scannen des entsprechenden Barcodes abgerufen und auf einem PDA oder Laptopbildschirm angezeigt werden.

Bestandteile der Arbeit

Ein System, das die genannten Szenarien ermöglicht, muss sowohl Autoren, bei der Zuordnung der Online-Ressourcen zu einem Dokument, als auch Leser, bei der Abbildung von Identifikationsnummern auf Online-Inhalte, von Dokumenten unterstützen.

Ein elementarer Dienst zur Unterstützung des Autors könnte als Druckertreiber realisiert sein, der beim Drucken eines Dokumentes automatisch eine Online-Version zusammen mit Metainformationen, wie Angaben zu Verfasser oder Druckzeitpunkt, erstellt und einen Barcode in das zu druckende Dokument einbettet. Dieser Minimalansatz kommt weitgehend ohne Eingreifen des Autors aus und stellt damit sicher, dass auch Informatik-Laien das System benutzen können und dass die Benutzung keinen Mehraufwand erfordert.

Für spezielle Metainformationen, wie der Gültigkeitsdauer eines Dokumentes oder bestimmten Aktionen, die beim Scannen der Markierung ausgeführt werden sollen, soll zusätzlich ein webbasiertes Werkzeug angeboten werden, das die Eingabe einer Dokumentbeschreibung und der dazugehörigen Online-Ressourcen ermöglicht. Dieser Dienst gibt daraufhin einen Barcode in verschiedenen Formaten zurück und legt die Dokumentbeschreibung in strukturierter Form, z.B. als XML-Dokument, ab.

Für die Leser von Dokumenten soll eine Infrastruktur entwickelt werden, welche die Abbildung von gelesenen Barcodes auf virtuelle Inhalte und Aktionen ermöglicht. Nach dem Einlesen der Barcodes können die damit verknüpften Online-Ressourcen eingesehen und entsprechende Aktionen auf dem Gerät des Benutzers (z.B. Anzeigen von Metainformationen, Erstellen von Kalendereinträgen) ausgelöst werden. Optional sollen auch die Leser eines Dokuments in der Lage sein, eigene Online-Inhalte zu ergänzen.

Ziele

- Entwicklung eines Dienstes zur Einbettung von Barcodes in Dokumente (a) als Druckertreiber und (b) als webbasiertes Werkzeug ("Tag Creation-Service")

- Erstellung eines XML-Schemas für verknüpfte Informationsinhalte und Metainformationen zu gedruckten Dokumenten
- Konzeption eines flexiblen Zuordnungsschemas von Online-Funktionalität zu Dokumenten
- Verwaltung der Online-Ressourcen zu gedruckten Dokumenten
- Umsetzung von gescannten Identifikationsnummern auf Online-Inhalte ("Mapping-Service")
- Benutzungskonzepte für das "Einsammeln" von Identifikationsnummern und der Anzeige von Inhalten auf Client-Geräten unter Berücksichtigung von verschiedenen zugrundeliegenden Hardwareplattformen (Memory-Barcodescanner, drahtlos kommunizierende Barcodescanner, PDAs, Workstations)
- Prototypische Erprobung einfacher Szenarien
- Skizzierung einer Erweiterung auf beliebige physische Objekte

Betreuung

Die Arbeit wird von Jürgen Bohn <bohn@inf.ethz.ch> und Michael Rohs <rohs@inf.ethz.ch> betreut und steht unter der Aufsicht von Prof. Dr. Friedemann Mattern.

Termine

Beginn der Diplomarbeit ist Montag, der 5. November 2001. Die schriftliche Ausarbeitung muss bis Montag, den 4. März 2002 eingereicht werden.

Prof. Dr. F.Mattern
Fachgruppe Verteilte Systeme
ETH Zürich

Anhang B

Glossar

DOM **D**ocument **O**bject **M**odel ist eine plattformneutrale und sprachenunabhängige Schnittstelle, um dynamisch auf (XML-)Dokumente zuzugreifen und deren Struktur und Inhalt zu ändern. Es ist eine offizielle Empfehlung des World Wide Web Consortiums (W3C). Ein DOM-Parser wandelt ein Dokument in eine Baumstruktur um und traversiert resp. manipuliert sie.

DTD **D**ocument **T**ype **D**efinition beschreibt, wie (XML-)Dokumente aussehen müssen, um gültig zu sein. Es beschreibt die Reihenfolge, Art, Kardinalität und Wertebereich von (XML-)Elementen des Dokuments.

EAN Das **E**uropean **A**rticle **N**umbering System ist der internationale Barcode-Standard für Produkte.

HTML **H**yper**T**ext **M**arkup **L**anguage ist eine auszeichnende Beschreibungssprache für Dokumente. Dokumente im HTML-Format werden in Browsern im vom Autor gewünschten Erscheinungsbild angezeigt.

RFID **R**adio **F**requency **ID**entification ist eine Form der drahtlosen Identifikation. RFID-Tags werden an Objekte angebracht, um sie dadurch zu identifizieren.

SAX steht für **S**imple **A**PI for **X**ML. Beim Parsen eines XML-Dokuments mittels SAX treten Events auf, die von einer Applikation verarbeitet werden können. SAX ist ein De-facto-Standard und eine alternative Möglichkeit, mit Inhalten eines XML-Dokuments zu arbeiten.

UPC ist ein Barcode-Typ und steht für **U**niversal **P**roduct **C**ode. Er wurde ursprünglich entwickelt um die Lagerbewirtschaftung zu beschleunigen und Prozessflüsse besser beobachten zu können. UPC wird in den USA eingesetzt.

XML E**X**tensible **M**arkup **L**anguage ist eine Strukturbeschreibungssprache auf logischer nicht auszeichnender Ebene und eignet sich deshalb sehr gut für Datenaustausch und -abgleich. XML ist eine Empfehlung des World Wide Web Consortiums (W3C).

XSLT E**X**tensible **S**tylesheet **L**anguage **T**ransformations ist ein standardisierter Mechanismust, wie ein XML-Dokument in ein anderes Dokument mit neuer Struktur umgewandelt werden kann. XSLT ist eine Empfehlung des World Wide Web Consortiums (W3C).

Anhang C

Eingesetzte Softwareprodukte

Dieser Anhang beschreibt die eingesetzten Programme, deren Konfiguration und allfällige Bugs, die gefunden wurden. Alle Softwareprodukte mit Ausnahme des Betriebssystems und der Programmiersprache befinden sich im Ordner "da-deploy" auf der Diplomarbeits-CD.

C.1 Windows 2000

Es wurde auf einem Windows 2000-Rechner gearbeitet. Als Dateieinsystem wurde NTFS benutzt, was wichtig für den Betrieb der Datenbank ist (siehe Abschnitt 7.7).

C.2 JAVA

Für den Betrieb von ETHOC muss Java auf Server-Seite als auch auf Klienten-Seite, sofern der ETHOC-Browser erwünscht ist, installiert sein. Für diese Diplomarbeit wurde JDK 1.3.1_01 eingesetzt.

C.3 Jakarta Tomcat

Als Serlvet-Engine wurde Jakarta Tomcat 4.0.1 [31] eingesetzt. Die Zip-Datei wurde ins Verzeichnis "C:\jakarta-tomcat-4.0.1" entpackt. Die Umgebungsvariable "CATALINA_HOME" wurde auf "C:\jakarta-tomcat-4.0.1" und die Umgebungsvariable"JAVA_HOME" auf "C:\dev\jdk1.3.1_01", den Pfad des Java Developer Kits, gesetzt.

Jakarta Tomcat wird mit dem Befehl "startup" gestartet. Die entsprechende Batch-Datei befindet sich unter "C:\jakarta-tomcat-4.0.1\bin\". Ist Tomcat Jakarta gestartet, können keine Java-Klassen mehr ausgetauscht oder eingefügt

werden. Die Installation der ETHOC-Distribution muss also vor dem Start erfolgen. Dies geschieht indem der Ordner “ethoc” (auf der CD im Ordner “da-deploy\webapps”) nach “C:\jakarta-tomcat-4.0.1\webapps\” kopiert wird.

Bugs

In dieser Arbeit wird WMLScript benutzt. WMLScript hat den Content-Typ “vnd.wap.wmlscript” Leider ist dieser Content-Typ in jakarta falsch gesetzt, was viele Mobiltelefone zum Abbrechen bewegt. Im Ordner “C:\jakarta-tomcat-4.0.1\conf\” befindet sich die Datei “web.xml”. Diese bestimmt unter anderem, welche Datei-Endungen welchen Content-Typ haben. Es muss “vnd.wap.wmls” durch
“vnd.wap.wmlscript” ersetzt werden.

C.4 XALAN

Zur XSL-Transformation wurde XALAN [25] eingesetzt. Die entsprechende Bibliothek “xalan.jar” befindet sich im Ordner “da-deploy” und muss ins Verzeichnis C:\jakarta-tomcat-4.0.1\common\lib kopiert werden.

C.5 XERCES

Xerces [26] ist ein XML-Parser und bereits in Jakarta Tomcat vorhanden. Es muss somit nichts installiert werden.

C.6 MySQL

Als Datanbank wurde MySQL [27] in der Version 3.23.46a verwendet. In der Zip-Datei befindet sich “Setup.exe”. Das Programm startet die Installation. MySQL wurde direkt unter dem Laufwerk C installiert. Die Datenbank wurde angepasst und um die ETHOC-Tabellen erweitert, indem MySQL mit dem Befehl “C:\mysql\bin\mysql -u root -p < install.sql.txt” gestartet wurde. Dabei wird auch das Root-Passwort gesetzt, welches zuvor leer war und einige Benutzer entfernt, die in der Standard-Konfiguration praktisch Root-Rechte haben.

MySQL muss mit der Option --set-variable max_allowed_packet=16M gestartet werden. 16 MByte ist laut MSQL-Manual [28] die maximale Paketgrösse des MySQL-Protokolls. Für die Version 4.0 ist eine Grösse von 4 GByte vorgesehen.

C.7 MM.MySQL

Für MySQL stehen zwei JDBC-Bibliotheken zur Verfügung. In dieser Arbeit wird die freie Bibliothek MM.MySQL [29] verwendet.

Bugs

Folgendes Fehlverhalten wurde bei der Abfrage "SELECT MAX(id) FROM IdCoord" festegestellt. Ist die Tabelle leer liefert die Methode "getInt()" der Klasse "ResultSet" die Zahl 0. Dies ist legitim, wenn anschliessend die Methode "wasNull()" den Wert "true" zurückgibt. Leider wird "false" zurückgegeben. Aus diesem Grund kann nicht unterschieden werden, ob das Maximum 0 oder leer (null) ist. Das Problem wurde in dieser Arbeit umgangen, indem zuerst überprüft wird, ob die Tabelle leer ist und dann die eigentliche Anfrage durchgeführt wird.

C.8 COS

COS steht für com.oreilly.servlet und ist eine Bibliothek, die den Umgang mit Datei-Uploads via Servlets erleichtert. In dieser Arbeit wurde die Version vom 19. Juni 2001 benutzt.

Bugs und Einschränkungen

Ein Bug (oder Feature?) der Bibliothek ist, dass der Header, der die Dateien und die Parameter enthält, nur einmal geparst werden kann und bei einem zweiten Versuch eine Exception auslöst. Aus diesem Grund mussten das ETHOC-Servlet, welches für die Aktionen verantwortlich ist, leicht abgeändert werden, damit es einen zweiten Aufruf vermeidet.

Die maximale Grösse des ganzen Headers inkl. Dateien, die übertragen werden, muss beim Aufruf der Bibliothek bestimmt werden. Ist diese grösser als 2^{30} Byte also grösser als 1GB, wird eine Exception geworfen.

C.9 SUNs JPEG-API

JPEG-Dateien werden mittels dem Paket "com.sun.image.codec.jpeg" [22] erzeugt. Da es Bestandteil von Java ist, muss es nicht extra installiert werden.

C.10 Acme Labs GIFEncoder

GIF-Dateien werden mit Hilfe der Klasse "Acme.JPM.Encoders.GifEncoder" [23] erzeugt. Um nicht das ganze Paket von Acme Labs zu installieren, wurden die verwendeten Klassen in die ETHOC-Hierarchie eingebaut und befinden sich unter "Ethoc.Authoring.Encoder". Somit sind die Klassen Bestandteil von ETHOC und müssen nicht speziell installiert werden.

C.11 SmtpClient

“sun.net.smtp.SmtpClient” ist Bestandteil aber keine Standard-Bibliothek von Java. Wird ein anderes Betriebssystem für ETHOC verwendet, muss die Funktionalität dieser Klasse zuerst überprüft werden, denn SUN gibt keine Garantien, dass die Klasse auf allen Plattformen existiert und funktioniert. Um Emails zu senden, muss unter anderem ein SMTP-Server angegeben werden. Es wird der SMTP-Server des Departements Informatik benutzt. Bei einer Installation muss beachtet werden, dass ein Server genommen wird, der die Anfragen von ETHOC akzeptiert.

Anhang D

XML-Definitionen

D.1 doc.dtd

In diesem Abschnitt wird die Document Type Definition eines virtuellen Dokumentes spezifiziert

```
<!ELEMENT ethoc (virt_doc)>
<!ATTLIST ethoc type (doc) #REQUIRED>

<!ELEMENT virt_doc (author, doc_info, contacts?, extend)>
<!ATTLIST virt_doc
    id CDATA #REQUIRED
    last_change CDATA #REQUIRED>

<!ELEMENT author (name, unit, office, email, phone, fax, homepage)>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name visible (y | n) #REQUIRED>

<!ELEMENT unit (#PCDATA)>
<!ATTLIST unit visible (y | n) #REQUIRED>

<!ELEMENT office (#PCDATA)>
<!ATTLIST office visible (y | n) #REQUIRED>

<!ELEMENT email (#PCDATA)>
<!ATTLIST email visible (y | n) #REQUIRED>

<!ELEMENT phone (#PCDATA)>
<!ATTLIST phone visible (y | n) #REQUIRED>

<!ELEMENT fax (#PCDATA)>
<!ATTLIST fax visible (y | n) #REQUIRED>

<!ELEMENT homepage (#PCDATA)>
<!ATTLIST homepage visible (y | n) #REQUIRED>
```

```
<!ELEMENT contacts (contact+)>
<!ELEMENT contact (name, unit, office, email, phone, fax, homepage)>

<!ELEMENT doc_info (title, keywords?, intro?, valid_until )>

<!ELEMENT title (#PCDATA)>

<!ELEMENT keywords (#PCDATA)>

<!ELEMENT intro (#PCDATA|br)*>

<!ELEMENT valid_until (year, month, day)>

<!ELEMENT extend (online_copy?, calendar?, modules, links?, files?)>

<!ELEMENT online_copy (doc_copy+)>

<!ELEMENT doc_copy (#PCDATA)>
<!ATTLIST doc_copy
    type CDATA #REQUIRED
    size CDATA #REQUIRED
    last_change CDATA #REQUIRED>

<!ELEMENT calendar (event+)>

<!ELEMENT event (summary, start, end, location, description)>
<!ATTLIST event
    last_change CDATA #REQUIRED
    time_zone CDATA #REQUIRED>

<!ELEMENT summary (#PCDATA)>
<!ELEMENT start (year, month, day, hour, min, sec)>
<!ELEMENT end (year, month, day, hour, min, sec)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT day (#PCDATA)>
<!ELEMENT hour (#PCDATA)>
<!ELEMENT min (#PCDATA)>
<!ELEMENT sec (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT description (#PCDATA|br)*>

<!ELEMENT modules (module+)>
<!ATTLIST modules last_config CDATA #REQUIRED>
<!ELEMENT module EMPTY>
<!ATTLIST module
    name CDATA #REQUIRED
    label CDATA #REQUIRED
    id CDATA #REQUIRED
    enabled (y | n) #REQUIRED
    link CDATA #IMPLIED
    last_change CDATA #REQUIRED>

<!ELEMENT links (link+)>
```

```
<!ELEMENT link (url, label)>
<!ATTLIST link last_change CDATA #REQUIRED>

<!ELEMENT url (#PCDATA)>
<!ELEMENT label (#PCDATA)>

<!ELEMENT files (file+)>
<!ELEMENT file (fname, label, publish_date)>
<!ATTLIST file
    type CDATA #REQUIRED
    size CDATA #REQUIRED
    last_change CDATA #REQUIRED>
<!ELEMENT publish_date (year, month, day)>

<!ELEMENT fname (#PCDATA)>

<!ELEMENT br EMPTY >
```

D.2 modules.dtd

Welche Module dem System zur Verfügung stehen, wird in der Datei "modules.xml" festgehalten, welche folgender Definition entsprechen muss:

```
<!ELEMENT modules (module+)>
<!ATTLIST modules last_config CDATA #REQUIRED>

<!ELEMENT module (name, label, entry_point_config, entry_point_use?,
notifier?)>
<!ATTLIST module
default_enabled (y | n) #REQUIRED
    inserted_by CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT entry_point_config (#PCDATA)>
<!ELEMENT entry_point_use (#PCDATA)>
<!ELEMENT notifier (#PCDATA)>
```

D.3 authoringtools.dtd

Welche Autorenwerkzeuge vorhanden sind, wird in der Konfigurationsdatei "authoringtools.xml" festgehalten, welche folgende Struktur aufweisen muss:

```
<!ELEMENT authoring (tool+)>
<!ELEMENT tool (name, entry_point)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT entry_point (#PCDATA)>
```

Literaturverzeichnis

[1] Friedemann Mattern, *The Vision and Technical Foundations of Ubiquitous Computing*, October 2001,
http://www.inf.ethz.ch/vs/publ/papers/ubicompvision.pdf

[2] Friedemann Mattern, *Ubiquitous Computing – Der Trend zur Informatisierung und Vernetzung aller Dinge*, In: Gerhard Rossbach (Hrsg.): Mobile Internet, Tagungsband 6. Deutscher Internet-Kongress, dpunkt-Verlag, September 2001, pp. 107-119
http://www.inf.ethz.ch/vs/publ/papers/Internetkongress.pdf

[3] Marc Langheinrich, Friedemann Mattern, Kay Römer, Harald Vogt, *First Steps Towards an Event-Based Infrastructure for Smart Things*, Ubiquitous Computing Workshop (PACT 2000), October 15-19, 2000, Philadelphia, PA.
http://www.inf.ethz.ch/vs/publ/papers/firststeps.pdf

[4] Jürgen Bohn, Michael Rohs, *Klicken in der realen Welt*, Konferenz Mensch und Computer 2001, Workshop Mensch-Computer-Interaktion in allgegenwärtigen Informationssystemen, Bad Honnef, 5.-8. März 2001
http://www.inf.ethz.ch/vs/publ/papers/mc2001.pdf

[5] Deborah Anglin, *Code Scanners and the Web*,
http://www.personal.psu.edu/users/a/a/aad152/6742/emerging.html

[6] T.L. Ashford, *Barcode Symbologies* –
http://www.zebra.com/about/aboutbar.htm

[7] Russ Adams *BarCode 1 – A Web Of Information About Bar Code*,
http://www.adams1.com/pub/russadam/

[8] Marshall Brain, *How UPC Bar Codes Work*,
http://www.howstuffworks.com/upc.htm/printable

[9] Gerhard Donalies, *Die Zukunft der automatischen Identifikation*, Fraunhofer-Magazin 2.1999,
http://www.fraunhofer.de/german/publications/df/df1999/299-11.htm

[10] Kevin Bonsor, *How Smart Labels Will Work*,
http://www.howstuffworks.com/smart-label.htm/printable

[11] CollTown-Projekt,
http://cooltown.hp.com

[12] AAES Research Studying Effects of Artificial Reefs on Marine Life, http://www.ag.auburn.edu/aaes/information/highlights/summer00/reef.html

[13] Tim Kindberg, Sandro Hawke, *The 'tag' URI scheme and URN namespace*, September 2001, http://search.ietf.org/internet-drafts/draft-kindberg-tag-uri-01.txt

[14] EAN Scheiz (www.ean.ch), *Barcoding: Technologien: UCC/EAN-128*, http://194.209.37.59/deutsch/04_Komponenten/01_Barcoding/04-ean128.html

[15] FinePrint, http://www.context-gmbh.de/fineprint.htm

[16] Doug Tidwell: *Introduction to XML*, Tutorial, Juli 1999 http://www.ibm.com/developerworks/education/xmlintro/

[17] Doug Tidwell: *XML programming in Java*, Tutorial, September 1999 http://www.ibm.com/developerworks/education/xmljava/

[18] Netscape, *Persistent Client State – HTTP Cookies, Preliminary Specification*, 1999, http://home.netscape.com/newsref/std/cookie_spec.html

[19] D. Kristol, L. Montulli *HTTP State Management Mechanism*, Februar 1997, http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2109.html

[20] Versit Consortium, *vCalendar – The Electronic Calendaring and Scheduling Exchange Format*, Version 1.0, September 1996, http://www.imc.org/pdi/vcal-10.ps

[21] Jason Hunter, *COS-Bibliothek*, Juni 2001 http://www.servlets.com/cos/index.html

[22] Sun Microsystems, *JPEG API*, http://java.sun.com/products/jdk/1.2/docs/guide/2d/api-jpeg/

[23] Acme Labs, *GIF Encoder*, http://www.acme.com/java/software/Acme.JPM.Encoders.GifEncoder.html

[24] Sun Microsystems, *JavaMail(TM) API*, Dezember 2000 http://java.sun.com/products/javamail/

[25] XALAN-Homepage, http://xml.apache.org/xalan-j/index.html

[26] XERCES-Homepage, http://xml.apache.org/xerces2-j/index.html

[27] MySQL-Downloadseite, http://www.mysql.org/downloads/index.html

[28] MySQL Reference Manual,
http://www.mysql.org/Downloads/Manual/manual.pdf

[29] MM.MySQL-Homepage, *A Type IV JDBC driver for MySQL*,
http://mmmysql.sourceforge.net/

[30] Mobileinfo, *Wireless Application Protocol - WAP*,
http://www.mobileinfo.com/WAP/components.htm

[31] Homepage Jakarta Tomcat,
http://jakarta.apache.org/tomcat/index.html