Diplomarbeit

Design und erweiterte Implementierung eines Datenbank Middleware Systems sowie eines Benutzerinterfaces für mobile Endgeräte im Rahmen des PROBADO Projektes

Tanja Reiterer

Institute of ComputerGraphics and KnowledgeVisualisation, TU Graz

www.cgv.tugraz.at

Kurzfassung

Im Zuge der zunehmenden Verbreitung digitaler Bibliotheken gewinnt auch die bibliothekarische Erfassung nicht textueller Objekte eine immer größere Bedeutung. Um die Wiederauffindbarkeit der gespeicherten Objekte und somit den Zugang zu diesen sicherzustellen, kommt der automatischen Generierung von Metadaten während des Einpflegevorgangs eine wesentliche Bedeutung zu. Im Rahmen des PROBADO Projektes wird nicht nur versucht, die gesamte bibliothekarische Prozesskette umzusetzen, sondern auch ein spezieller Focus auf die automatische Metadatenextraktion und den Aufbau einer gezielten Metadatensuche gelegt.

Das Projekt gliedert sich grob in drei Ebenen, die Ebene der Benutzerinterfaces, die Middleware (PROBADO Core), ein bereichsübergreifendes Metadaten Repository, über das sämtliche Suchanfragen abgewickelt werden, sowie die Ebene der Spezial Repositorien für nicht textuelle Objekte. Die vorliegende Arbeit gliedert sich in einen theoretischen und einen praktischen Teil, wobei im theoretischen Teil zunächst auf die Grundlagen digitaler Bibliotheken eingegangen wird, gefolgt von einem Vergleich existierender digitaler Bibliothekssysteme mit PROBADO. Weiters werden verschiedene Möglichkeiten zur Abfrage und Präsentation von Bibliotheksinhalten auf mobilen Endgeräten evaluiert und gegenübergestellt.

Inhalt des praktischen Teils dieser Arbeit sind zunächst Designanpassungen die in der letzten Projektphase an der CORE Schicht, sowie im 3D Bereich vorgenommen wurden, (Datenbankdesign, Schnittstellen) wobei der serviceorientierte Ansatz und ein Teil der Schnittstellenspezifikation beibehalten wurden. Darauf aufbauend erfolgte die Reimplementierung des CORE Prototypen unter Einsatz von C# und MS SQL-Server als Database Management System, sowie die Schaffung eines Benutzerinterface für mobile Endgeräte für PROBADO 3D.

Abstract

In the course of the increasing spread of digital libraries the storage and retrieval of non textual objects becomes increasingly important. To reassure the possibilty of stored object retrieval automatic generation of metadata during the storage process is significant. The PROBADO project not only tries to implement the process chain of a library but gives special attention to automatic extraction of metadata and the implementation of specialised metadata search. The project is roughly divided into three layers, the user interface layer, the middleware layer (PROBADO Core, an inter-divisional metadata repository) and the repository layer, that contains specialised repositories for non textual objects.

This thesis is divided into a theoretical and a practical part. The theoretical part concentrates on the fundamentals of digital libraries, followed by a comprison of exisiting digital library management systems with PROBADO. Finally an evaluation of options of accessing and presenting library content on mobile devices was carried out.

The practical part explains the CORE and 3D design changes implemented in the last project phase (database, interfaces). The service oriented design and part of the interface specification have been kept. The reimplementation of the CORE prototype was done using C# and MS SQL-Server. Furthermore a user interface for access to PROBADO from mobile devices was created.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, am 13. April 2013

(Tanja Reiterer)

Danksagung

An dieser Stelle möchte ich mich bei Dr. Tobias Schreck und Dr. Sven Havemann für die Betreuung während der Erstellung meiner Masterarbeit bedanken. Ebenfalls möchte ich den Mitgliedern des PROBADO-Projektteams, insbesondere Herrn Dipl.-Inf. Maximilian Scherer, Herrn Dipl.-Inf. René Berndt und Frau Dipl.-Ing. Ina Blümel, für die gute Zusammenarbeit danken, sowie meinen Arbeitskollegen bei AFGA und meinem Vorgesetzten Dipl.-Ing. Stefan Leiteritz für die Unterstützung. Ganz besonderer Dank geht an meinen Mann Emanuel Reiterer.

Graz, am 13. April 2013 Tanja Reiterer

Inhaltsverzeichnis

1 Einleitung **1**

2 Begriffsdefinitionen **3**
2.1 XML . . . 3
2.2 Metadaten . . . 3
2.2.1 MARC . . . 4
2.2.2 Dublin-CORE . . . 4
2.2.3 METS . . . 5
2.2.4 CIDOC CRM . . . 5
2.2.5 OAI-PMH . . . 5
2.2.6 RDF . . . 6
2.3 SOA . . . 6
2.4 SOAP . . . 7
2.4.1 Aufbau einer SOAP Nachricht . . . 7
2.5 REST . . . 8
2.6 WSDL . . . 9
2.7 OpenSearch . . . 9

3 Verwandte Arbeiten **11**
3.1 Begriffsdefinitionen . . . 11
3.1.1 Digitale Bibliothek (DL) . . . 11
3.1.2 Digitales Bibliothekssystem (DLS) . . . 11
3.1.3 Digitales Bibliotheks Management System (DLMS) . . . 11
3.2 Ein Framework in drei Schichten . . . 12
3.3 Kernkonzepte zur Definition einer Digitalen Bibliothek . . . 12
3.3.1 Inhalt . . . 12
3.3.2 Benutzer . . . 13
3.3.3 Funktionalität . . . 13
3.3.4 Qualität . . . 13
3.3.5 Richtlinien . . . 14
3.3.6 Architektur . . . 14
3.3.7 DLS Zusammenhang zwischen den Kernkonzepten . . . 15
3.4 DL Ressourcenmodell . . . 15
3.4.1 Content Domain . . . 15
3.4.2 Informationsobjekt . . . 16
3.4.3 Metadaten . . . 16
3.5 Semantische Digitale Bibliotheken . . . 17
3.5.1 Konzept . . . 17
3.5.2 Knowledge Organisation System (KOS) . . . 18
3.5.3 Ontologie . . . 19
3.5.4 Soziale Semantische Informationsräume . . . 20
3.5.5 Ziele einer Semantischen Digitalen Bibliothek . . . 21
3.5.6 Architekturkonzepte einer Semantischen Digitalen Bibliothek . . . 21

4 Beispiele für DLMS **24**
4.1 DELOS DLMS . . . 24
4.2 BRICKS . . . 24
4.3 DILIGENT . . . 25
4.4 MARIAN . . . 26
4.5 Fedora . . . 27
4.6 REAP . . . 27
4.7 EUROPEANA . . . 28
4.8 PROBADO . . . 29
4.8.1 Inhaltsbasierte Erschließung und Ergebnispräsentation . . . 31
4.9 Vergleich der Systeme . . . 31
4.9.1 Dokumentenformate . . . 32
4.9.2 Systemarchitektur . . . 33
4.9.3 Schnittstellen/Protokolle . . . 33
4.9.4 Programmiersprache . . . 33
4.9.5 Unterstützte Plattformen . . . 34
4.9.6 Projektbezogene Daten . . . 34
4.9.7 Zusammenfassung . . . 34

5 Eingesetzte Techniken **36**
5.1 Technologiewechsel . . . 36
5.2 Eingesetzte Technologien . . . 36
5.2.1 Microsoft Silverlight . . . 37
5.2.2 LINQ . . . 38
5.3 Entwicklungsumgebungen . . . 39

6 Umsetzung **41**
6.1 Ausgangslage und Zielsetzung . . . 41
6.2 Aufbau von PROBADO . . . 41
6.2.1 Frontend Layer . . . 42
6.2.2 Core Layer . . . 44
6.2.3 Repository Layer . . . 44
6.3 Metadaten im CORE . . . 46
6.4 Datenbankmodell . . . 47
6.4.1 Allgemeine Stammdaten . . . 47
6.4.2 Repositorien Datenmodell . . . 48
6.4.3 Metadatenmodell . . . 49
6.4.4 Administratives Modell . . . 51
6.4.5 Struktur zur Protokollierung von Suchabfragen . . . 52
6.5 Personalisierung und Benutzerfeedback . . . 56
6.5.1 Motivation . . . 56
6.5.2 Benutzerfeedback . . . 56
6.5.3 Personalisierung . . . 56
6.5.4 Statistische Auswertung . . . 56
6.5.5 Protokollierung der Suchabfragen in PROBADO . . . 56
6.6 Webservices . . . 57
6.7 Repository Registrierung . . . 58
6.7.1 Metadatensatz Repository Registrierung . . . 58
6.7.2 Metadatensatz Queryengine Registrierung . . . 58
6.7.3 Ablauf Repository Registrierung . . . 58
6.7.4 SOAP Nachrichten Repository Registrierung . . . 59
6.7.5 Änderung von Repository Daten . . . 60
6.8 Metadatenaustausch . . . 61
6.8.1 Vorgehen beim Export . . . 61
6.9 Suchanfragen . . . 64
6.9.1 Allgemein . . . 64
6.9.2 ProbadoSearch . . . 64

6.9.3 CORE Metadatensuche . . . 67
6.9.4 CORE Inhaltsbasierte Suche . . . 67
6.9.5 Ergänzende Schnittstellenfunktionen CORE Suche . . . 68
6.10 RepositorySearch . . . 69

7 Datenbankanpassungen im 3D Bereich 70
7.1 Gestaltung eines neuen Datenbankmodells . . . 70
7.1.1 Schwachstellen des ursprünglichen Datenbankmodells . . . 70
7.2 Vorgangsweise . . . 70
7.2.1 Überarbeitung Datenbankmodell . . . 72
7.3 Mehrsprachigkeit . . . 72
7.3.1 Umsetzung im Datenbankschema . . . 72
7.3.2 Volltextsuche . . . 73
7.3.3 Export und Import bestehender Daten . . . 73

8 Benutzerinterface für mobile Endgeräte 74
8.1 SOAP Unterstützung auf mobilen Plattformen . . . 75
8.1.1 Android . . . 75
8.1.2 Windows Phone 7 . . . 75
8.1.3 iOS . . . 76
8.2 Darstellung von 3D Objekten auf mobilen Endgeräten . . . 76
8.2.1 X3DOM . . . 77
8.2.2 OpenGL . . . 78
8.2.3 NinevehGL . . . 78
8.2.4 DirectX . . . 78
8.2.5 Vergleich der untersuchten APIs . . . 78
8.3 Zusammenfassung . . . 79
8.4 Probado 3D für Windows Phone 7 . . . 79
8.5 iProbado 3d . . . 79
8.5.1 Ablauf der Abfragen . . . 80
8.5.2 3D Darstellung in iOS . . . 80

9 Lessons Learned 88

10 Zusammenfassung 90
10.1 Schlussfolgerung . . . 90
10.2 Ausblick . . . 91

Bibliography 93

List of Tables 98

List of Figures 99

Index 101

Kapitel 1

Einleitung

„*I have always imagined that paradise will be a kind of library*“ sagt der Autor Jorge Luis Borges in einem seiner Werke. [Bor00] Ein Zitat, das sich auch auf einer Webseite des *DELOS*[1] Projekts (*Network of Excellence of Digital Libraries*) wiederfindet, eines Projekts der Europäischen Kommission mit dem Ziel, laufende Forschungsarbeiten im Bereich Digitaler Bibliotheken zu koordinieren und eine neue Generation von Technologien in diesem Bereich zu schaffen. [DEL10] Auch die im Rahmen des *PROBADO* Projektes entwickelten Technologien sind in diesem Kontext zu sehen.

Zunächst stellt sich die Frage, welche Eigenschaften die oben erwähnte Bibliothek auszeichnen. Diese sind das Vorhandensein und die Wiederauffindbarkeit von Wissen.

Im Zuge der Entwicklung moderner Informationstechnologien in den letzten Jahrzehnten kam es zu einer radikalen Änderung im Bibliothekswesen. Traditionelle Inhalte wie Bücher, Zeitschriften und Dokumente werden immer mehr durch digitale Inhalte ergänzt oder gar abgelöst. Diese digitalen Inhalte umfassen neben textuellen Inhalten ein weites Spektrum von Objekten wie z.B. Bilder, Grafikmodelle, Videos, Musik, E-Learning Objekte und viele andere mehr.

Für klassische textuelle Inhalte, die in digitaler Form vorliegen, sind bereits ausgereifte Mechanismen zur Integration in den Arbeitsablauf der Bibliotheken vorhanden. Texte können nahezu automatisch digitalisiert, indexiert und mit Metadaten versehen werden. Im Bereich multimedialer Daten (Bilder, Videos, Musik usw.) ist die Integration komplizierter. Diese Daten liegen in domänenabhängigen Formaten vor und sind schwierig zu indizieren, was die Suche nach ihnen erschwert. Dies macht die Entwicklung flexibler neuer Bibliotheksmanagementsysteme zur Unterstützung der inhaltsbasierten Ablage, Indizierung und Suche nach Multimediaobjekten erforderlich.

Ziel des PROBADO Projektes (*Prototypischer Betrieb Allgemeiner Dokumente)* ist es, den gesamten Workflow einer digitalen Bibliothek für nicht textuelle Dokumente zu unterstützen, beginnend mit der Erfassung und teils automatischen Indizierung von Dokumenten, bis hin zur Suche und Ergebnispräsentation. [BBC*10]

Es handelt sich dabei also um einen Bibliotheksdienst für allgemeine Dokumenttypen, der sich in der ersten Projektphase vor allem der Erschließung von Dokumenten aus den Bereichen 3D Computergrafik und Musik widmet. Dabei wurde besonderes Augenmerk auf die Gestaltung spezieller Interfaces für die Gewinnung inhaltsbasierter Metadaten, die inhaltsbasierte Suche und die Dokumentenpräsentation gelegt.

Das Projekt wurde im Jahr 2005 ins Leben gerufen und gliedert sich in drei Phasen. Diese Diplomarbeit leistet einen Beitrag zur dritten Projektphase. Im Rahmen der ersten beiden Projektphasen wurde zunächst eine Vorerhebung und in der zweiten Phase eine Studie der verschiedenen Möglichkeiten zur technischen Realisierung durchgeführt (Diplomarbeit Monika Alter [Alt09]). Das Projekt teilte sich zunächst in vier Teilbereiche (3D, Musik, E-Learning und CORE) und wurde nach dem Ausstieg des E-Learning Bereichs auf die verbleibenden Bereiche reduziert. Dabei setzten die einzelnen Projektpartner auf unterschiedliche Technologien.

Im Zuge der Studie von Monika Alter entstand ein Umsetzungskonzept für den CORE-Bereich, aus dem ein erster CORE-Prototyp hervorging, der aber noch nicht voll funktionsfähig war. Dieser Prototyp stützte sich auf das Open Source Framework *Apache Cocoon*[2] und eine *MySQL*[3] Datenbank. Im prak-

[1] http://www.delos.info/
[2] http://cocoon.apache.org/
[3] http://dev.mysql.com/

tischen Einsatz zeigten sich folgende Nachteile: Lange Einarbeitungszeiten für Mitarbeiter aufgrund der Komplexität und des Umfangs des Apache Cocoon Frameworks und Instabilitäten des bestehenden Sourcecodes aufgrund mangelnder Abwärtskompatibilität neuer Versionen von Apache Cocoon. Bei jedem Releasewechsel des Basisframeworks musste der gesamte Sourcecode angepasst werden um die bestehende Funktionalität wiederherzustellen. Das führte im Laufe der Zeit zu einer Ansammlung von schwer überschaubaren und kaum wartbaren Codebestandteilen. Erschwerend kam hinzu, dass für Cocoon keine integrierte Entwicklungsumgebung zur Verfügung stand, was die Fehlersuche sehr aufwändig gestaltete.

Aus diesen Gründen wurde beschlossen in den Bereichen 3D und CORE einen Technologiewechsel zugunsten einer kommerziellen Entwicklungsumgebung (*Microsoft Visual Studio*[4]) zu vollziehen. Im Zuge dessen wurde auch MySQL durch *Microsoft SQL-Server*[5] als DBMS abgelöst.

Ausgelöst durch die zunehmende Verbreitung von Smartphones und Tablets und die steigende Nachfrage nach mobilen Applikationen wurde in der Endphase des Projektes beschlossen, die Möglichkeit eines Zugriffs auf PROBADO 3D über mobile Endgeräte zu schaffen.

Die vorliegende Arbeit beschäftigt sich mit der praktischen Umsetzung dieses Technologiewechsels sowie der Erstellung eines Benutzerinterfaces für mobile Endgeräte. Daraus ergaben sich drei Gesamtziele mit den zugehörigen Unterzielen.

1. Ein verbessertes 3D Datenbankmodell für Microsoft SQL-Server, welches Mehrsprachigkeit unterstützt.
   (a) Erweiterung des 3D Datenbankmodells
   (b) Umstellung von MySQL auf Microsoft SQL-Server
   (c) Portierung der bestehenden Daten
2. Ein funktionsfähiger Prototyp für PROBADO Core
   (a) Neugestaltung des DB-Schemas für PROBADO Core
   (b) Überarbeitung der *WSDL* [Sch10d] Protokolle für Suche und Datenaustausch mit PROBADO Core
   (c) Webservice zur Core Suche
   (d) Webservice für Datenabgleich mit Core
   (e) Erweiterung der Core Suche (Personalisierung)
3. Ein Benutzerinterface für mobile Endgeräte für PROBADO 3D
   (a) Vergleich vorhandener mobiler Endgeräte und Auswahl einer geeigneten mobilen Plattform
   (b) App für iPad/iPhone zum Zugriff auf das PROBADO 3D Webservice
   (c) 3D Vorschau mit OpenGL ES

Der weitere Aufbau der vorliegenden Arbeit ist wie folgt gestaltet:

Kapitel 2 enthält Begriffsdefinitionen. Kapitel 3 erklärt die theoretischen Grundlagen digitaler Bibliotheken, Kapitel 4 gibt einen Überblick über verwandte Arbeiten und vergleicht diese mit dem PROBADO Projekt. Kapitel 5 beschreibt die bei der praktischen Umsetzung eingesetzten Techniken. Kapitel 6 und 7 beinhalten eine genaue Beschreibung des Core-Prototypen sowie der Anpassungen im 3D-Bereich. Kapitel 8 befasst sich mit Design und Umsetzung der PROBADO 3D App. Kapitel 9 geht auf die während der Umsetzung gesammelten Erfahrungen ein und beschreibt die daraus gewonnenen Erkenntnisse. Kapitel 10 enthält eine Zusammenfassung der Ergebnisse und gibt einen Ausblick auf zukünftige mögliche Entwicklungen.

[4] http://www.microsoft.com/germany/visualstudio/
[5] http://www.microsoft.com/germany/sql/2008/default.mspx

Kapitel 2

Begriffsdefinitionen

In diesem Kapitel sollen einige Begriffe und Grundlagen, auf die in den folgenden Kapiteln eingegangen wird, näher erklärt werden. Dabei handelt es sich einerseits um den Bereich *XML*, *XML-basierte Sprachen*, *Metadaten* und *RDF*, sowie andererseits um das Themengebiet der *Service Oriented Architecture* (SOA) und in diesem Zusammenhang um Definitionen von Begriffen wie *SOAP*, *REST* und *WSDL*. Da auf einen Teil dieser Themenbereiche in der Diplomarbeit von Monika Alter [Alt09] bereits ausführlich eingegangen wurde, sind die meisten der folgenden Definitionen nur kurz zusammengefasst.

2.1 XML

XML (*Extensible Markup Language*) ist ein vom W3C Konsortium definierter Standard für die Repräsentation von Daten in elektronischer Form [HM02], basierend auf einem ausgewählten Teilbereich der älteren SGML (*Standard Generalized Markup Language*) Spezifikation, der speziell für die Verwendung im Internet geeignet ist [KR08]. Es handelt sich dabei um eine hierarchisch aufgebaute Metasprache. Sie unterstützt Unicode und wird vor allem für den Austausch von Daten in textueller (lesbarer) Form eingesetzt. XML selbst wird meist in Verbindung mit verwandten Technologien wie *XPath*, *XSLT*, *Xquery* und *XML Schema Sprachen* eingesetzt. Eine Vielzahl anwendungsspezifischer Sprachen wie zum Beispiel XHTML oder WSDL basieren auf XML. [Ril10]

2.2 Metadaten

Die klassische Definition von *Metadaten* lautet "*Daten über Daten*". Sie werden dazu verwendet, unterschiedliche Aspekte von Daten wie Inhalt, Qualität, Bedingungen und andere Charakteristiken auszudrücken. Diese Daten können in verschiedenen Kontexten und für verschiedene Zwecke eingesetzt werden. Sie können folgendermaßen unterteilt werden [Lib10]:

- **Beschreibende Metadaten**
  Beinhalten Informationen zum Wiederauffinden und zur Beschreibung und Interpretation von Informationsobjekten
- **Administrative Metadaten**
  Daten, die das Repository für die Administration des Objektes benötigt. Darunter fallen technische Metadaten wie das Datenformat, Copyright und Lizenzinformationen, sowie notwendige Daten für die Langzeitarchivierung (erhaltende Metadaten)
- **Strukturelle Metadaten**
  Verbindende Information, die Zusammenhänge zwischen einzelnen Informationsobjekten herstellt.

Wie in der Oxford Digital Library beschrieben, spielen Metadaten im Zusammenhang mit digitalen Bibliotheken eine wesentliche Rolle. Sie ermöglichen es, zusätzliche Informationen zu den gespeicherten Objekten abzulegen, die diese charakterisieren und somit „auffindbar" machen. Aus diesem Grund wurde im Laufe der Zeit in existierenden digitalen Bibliothekssystemen, bei der Einpflege von Objekten ins System, eine immer größere Anzahl an Metadaten extrahiert und abgelegt. Zunächst vor allem in Bezug

auf Bücher, später in zunehmendem Maße für beliebige digitale Objekte (Videos, Architekturmodelle, Bilder, usw.). Um den Austausch dieser Daten zwischen den einzelnen Bibliotheken und Institutionen zu erleichtern, wurden Standards zur Definition von Struktur und Inhalt solcher Metadatensätze entwickelt. Einige dieser Standards werden im folgenden Abschnitt angeführt und kurz beschrieben. [Lib10]

2.2.1 MARC

Bei MARC (*machine-readable cataloging record*) handelt es sich um einen weitverbreiteten Standard zur Erzeugung von maschinenlesbaren Einträgen für Bibliothekskataloge. Dieser wurde in den 60-er Jahren des letzten Jahrhunderts von der Library of Congress entwickelt und wird in der einen oder anderen Variante von beinahe allen, heute genutzten Bibliothekskatalogen verwendet. Er ist sehr umfangreich und speziell auf die traditionellen Arbeitsweisen in klassischen Bibliotheken abgestimmt und ist daher nur sehr begrenzt für Bibliotheksobjekte, bei denen es sich nicht um Bücher handelt, einsetzbar. [Lib10]

2.2.2 Dublin-CORE

Bei *Dublin Core* handelt es sich um ein von der *Dublin Core Metadata Initiative* (DCMI) [Ini10] festgelegtes Set von Metadatenelementen zur Beschreibung von Ressourcen im Web. [KM09] Dublin Core besteht in der einfachen Variante aus 15 Elementen, welche Informationen über Inhalt, Urheber und formale Kriterien geben und in der erweiterten Variante aus drei zusätzlichen Elementen und einer Gruppe von Qualifikatoren, die zur Verfeinerung der einfachen Elemente verwendet werden können. [Trä00] Die Gruppe der Qualifikatoren lässt sich grob in zwei Bereiche einteilen, ist als Richtlinie anzusehen und offen für Erweiterungen [Ini11]:

- **Element Refinement.** Diese Gruppe von Qualifiern dient zur genaueren Spezifikation der Bedeutung eines Elements.
- **Encoding Scheme.** Diese Qualifier werden zur Identifikation zugeordneter Codierungs-Schemata eingesetzt. Ein solches Schema enthält z.B. Listen vorgegebener Werte, Darstellungs-und Formatierungsregeln für beispielweise Datumsformate usw.

Dublin Core ist so allgemein gehalten, dass es zur Beschreibung beliebiger Objekte verwendet werden kann. Der Anwender wird nicht dazu gezwungen, beim Einsatz von Dublin-CORE Elementen eine bestimmte Syntax einzuhalten. Alle Felder sind optional und können mehrfach und in beliebiger Reihenfolge verwendet werden. [Som04]

Vorteile Die Vorteile von Dublin Core liegen in seiner Einfachheit, der Flexibilität und individuellen Erweiterbarkeit und in der universellen Einsetzbarkeit. Seine Handhabung ist leicht zu erlernen, und er kann sprachunabhängig und bereichsübergreifend eingesetzt werden. Er ist weit verbreitet und wird von vielen digitalen Bibliothekssystemen unterstützt.

Nachteile Dublin Core wurde als kleinster gemeinsamer Nenner zu Kategorisierung verschiedenster Bereiche geschaffen, was seine Handhabung zwar sehr vereinfacht, aber durch das fehlende Regelwerk zu mangelhafter Interoperabilität und Problemen bei der Abbildung komplexer Zusammenhänge führt. [Lag00] Vor allem im Bereich des Weltkulturerbes bestehen oft komplexe semantische Beziehungen zwischen den einzelnen Objekten. Als Beispiele wären hier Verwandschaftsbeziehungen zwischen Künstlern zu nennen (z.B. "*zeige mir alle Bilder, die von den Söhnen Pieter Breughels gemalt wurden*" , oder das bekannte von Doerr veröffentlichte Beispiel von verschiedensten Objekten, wie Bildern, Dokumenten, Texten, Filmaufzeichnungen etc., die alle im Umfeld eines Ereignisses, in diesem Fall der Konferenz von Jalta, entstanden sind. Um diese Beziehungen darzustellen, muss der qualifizierte Dublin Core um eine Vielzahl von optionalen Attributen erweitert werden, was wiederum zu Problemen im Datenaustausch führt, da die Gegenseite diese Attribute entweder ignoriert, was zu einem kompletten Datenverlust führt, oder mühsam die Schnittstelle manuell anpassen muss. [Doe03]

Da in Webseiten eingebaute Meta-Angaben von den gängigen Suchmaschinen bereits seit einiger Zeit vollkommen ignoriert werden, kann die Einbindung von Dublin Core Elementen als Meta-Tags maximal im internen Bereich als sinnvoll angesehen werden. [SEL11]

2.2.3 METS

METS (*Metadata Encoding & Transmission Standard*) ist ein von der Kongressbibliothek entwickelter und gepflegter, XML-basierter Standard, um zu digitalen Bibliotheksobjekten beschreibende, administrative und strukturelle Metadaten abzulegen und einen Repository übergreifenden Austausch zu ermöglichen. Seine Entwicklung wurde 2001 durch eine Initiative der Vereinigung digitaler Bibliotheken angestoßen. [oC11] Der Standard ist im Bibliothekssektor weit verbreitet und als Metadaten-Container anzusehen, der beschreibende, administrative und strukturelle Metadaten enthalten kann, die sich wiederum auf andere Schemata beziehen können (z.B. MARC). Ein METS Dokument besteht aus sieben Sektionen [KM09]:

- **METS Header**: beinhaltet grundsätzliche Informationen zum Dokument (Autor, Entstehungsdatum usw.)
- **Beschreibende Metadaten**: beinhaltet oder referenziert Metadaten zum Dokument, die in verschiedenen Schemata vorliegen können.
- **Administrative Metadaten**: beinhaltet technische Informationen wie Fileformat, Kompression, Größe usw.
- **File-Bereich**: listet alle Dateien auf, die das digitale Objekt verwenden.
- **Struktureller Aufbau**: beschreibt die logische Struktur des digitalen Objektes.
- **Strukturelle Links**: listet sämtliche im strukturellen Aufbau eingesetzte Hyperlinks auf.
- **Verhaltensbereich**: beschreibt das Ausführungsverhalten von Teilen des Objektes.

METS ist wesentlich komplexer aufgebaut als Dublin Core und erfordert dadurch auch in der Einführungsphase einen viel höheren konzeptionellen Aufwand. Sein Einsatzgebiet ist vor allem im Bereich der Bibliotheken und Archive angesiedelt. [oI11]

2.2.4 CIDOC CRM

Das *CIDOC Conceptual Reference Model* bezeichnet einen Standard zur Modellierung von Informationen zum Kulturerbe über eine Ontologie. Ziel dieser Ontologie ist es, trotz unterschiedlicher Datenbank und Metadatenstrukturen Konzepte und Beziehungen zwischen kulturellen Elementen herzustellen. Die Basisontologie besteht aus einer Anzahl allgemein gehaltener Klassen wie z.B. Ort, Ding, handelnde Person, Zeitbereich und einer Reihe von Beziehungen, die zur Verknüpfung dieser Klassen eingesetzt werden, um so Inhalte und Zusammenhänge aufzuzeigen. [KM09] Die aus der aktuellen Definition von CIDOC CRM [Gro06] entnommene Abbildung 2.1 gibt hierzu ein Beispiel.

Sie zeigt wie zeitabhängige Informationen von CRM dargestellt werden. Das Diagramm enthält 4 Hauptklassen (E2 Temporal Entity, E52 Time-Span, E61 Persistent Item und E53 Place). Beziehungen zwischen den Hauptklassen und deren Subklassen werden in Form von Pfeilen abgebildet. Beispielsweise handelt es sich bei E2 Temporal Entity um eine abstrakte Klasse, von der sämtliche Klassen mit einer zeitlichen Komponente abgeleitet sind (Period, Event und Condition State). Erkennbar im Diagramm durch einen Doppelpfeil. Eigenschaften und Beziehungen zwischen Klassen werden durch einfache Pfeile dargestellt. E22 Time-Span hat beispielsweise die transitive Eigenschaft „falls within (contains)“. Beispielsweise kann die Zeitspanne der Errichtung eines Gebäudes in eine bestimmte historische Periode fallen. [Gro06]

Der große Vorteil des ontologischen Ansatzes im Gegensatz zu Metadaten Schemata wie Dublin Core liegt darin, dass sich mittels Ontologien auch komplexe Relationen zwischen Objekten, Eigenschaften und (Mehrfach-)Vererbungen darstellen lassen, die es ermöglichen logische Schlüsse zu ziehen und Objekte über übergeordnete Klassen zu identifizieren.

2.2.5 OAI-PMH

Das *OAI Protokoll für Metadata Harvesting*, kurz OAI-PMH wurde von der Open Archives Initiative entwickelt und liegt aktuell in der Version 2.0 vor. Es definiert einen Mechanismus für das Sammeln von Metadaten aus unterschiedlichen Repositorien. Es basiert auf den Standards http und XML. Die gesammelten Metadaten können in beliebiger Form vorliegen, jedoch wird ein einfaches Dublin-CORE

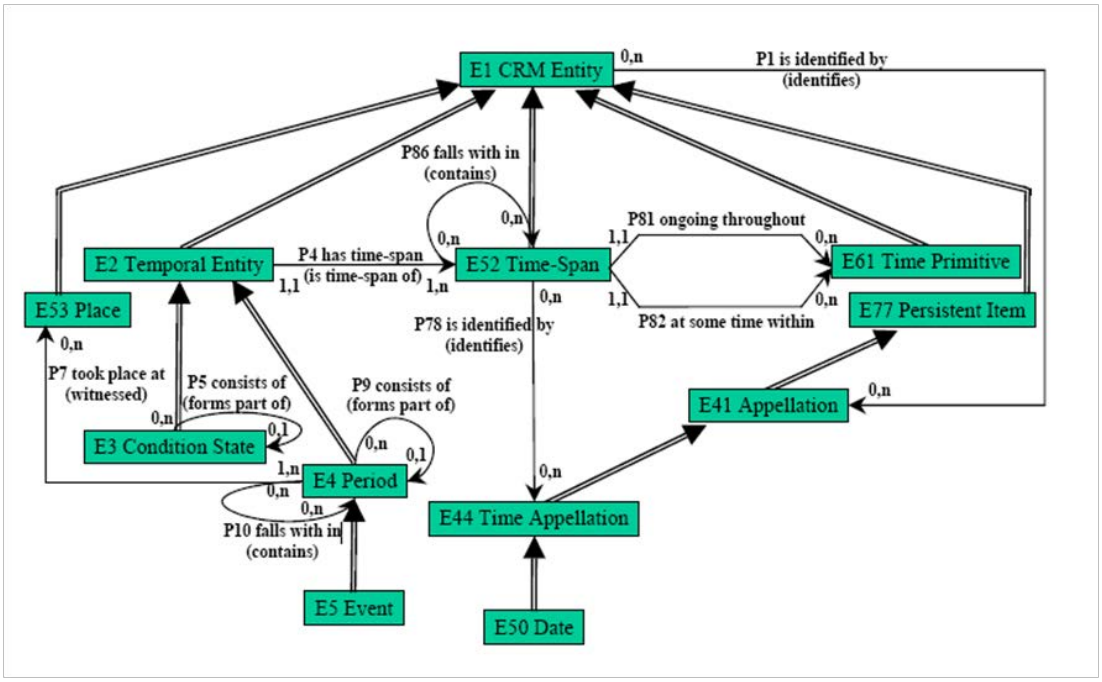


Abbildung 2.1: Abbildung von zeitbezogenen Informationen durch CRM. Quelle: [Gro06]

Format empfohlen. Die Funktionalität des OAI Protokolls beschränkt sich auf das Sammeln der Daten und muss zur Implementierung eines Suchmechanismus innerhalb der Metadaten mit anderen Mechanismen kombiniert werden. [For10]

2.2.6 RDF

Unter RDF (*Ressource Description Framework*) versteht man ein standardisiertes Modell für den Datenaustausch im Web. Es ist ein XML-basiertes Metadaten Datenmodell. [Sch10b] Obwohl sein hauptsächliches Einsatzgebiet die Repräsentation von Webressourcen wie z.B. Titel, Entstehungsdatum oder Autor einer Webseite ist , kann es auch für die allgemeine Beschreibung von Objekten, die über das Web identifiziert werden können, verwendet werden. Beispielsweise für Artikel, die über einen Onlineshop bezogen werden können (Artikelbeschreibung, Seriennummer, Preis, Lieferbedingungen usw.). Die Grundidee von RDF liegt darin, Elemente im Web mittels eindeutiger Kennungen genannt „Uniform Ressource Identifier“ oder kurz URI zu identifizieren und Ressourcen als Kombinationen von Eigenschaften und Eigenschaftswerten darzustellen. Eine Ressource kann somit als Graph dargestellt werden, dessen Knoten und Kanten die Ressource sowie ihre Eigenschaften repräsentieren. [Sch10b]

Die Abbildung: 2.2, entnommen aus der W3C Webseite, definiert eine Person mit Namen Eric Miller mit dem Titel Dr. und der Email Adresse em@w3c.org identifiziert durch http://www.w3.org/People/EM/contact#me.

2.3 SOA

SOA steht für *Service Oriented Architecture* und ist *„ein Architekturkonzept, das die Bereitstellung fachlicher Dienste und Architekturen in Form von Services vorsieht. Ein Service ist in diesem Kontext eine Funktionalität, die über eine standardisierte Schnittstelle in Anspruch genommen werden kann“.* [BZ06]

Das Architekturprinzip sieht eine Menge unabhängiger, lose gekoppelter Dienste vor, die über das Anfrage und Antwortprinzip (*„service request“*, *„service response“*) miteinander kommunizieren. Komplizierte Prozesse werden so über eine Abfolge von Serviceaufrufen abgearbeitet. SOA basierte Applikationen werden oft als Webservices umgesetzt, wobei jede beliebige dienstebasierte Technologie eingesetzt werden kann. In der Praxis spielen hier die Standards SOAP, REST sowie WSDL eine wichtige

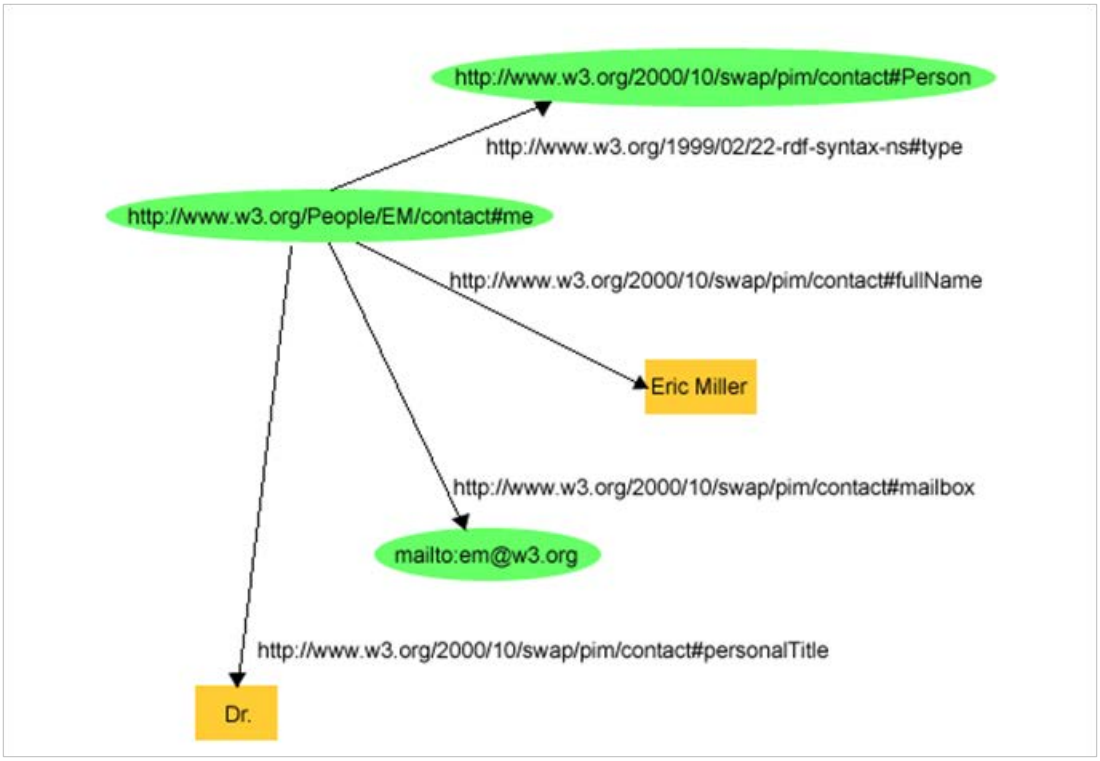


Abbildung 2.2: RDF Graph für Eric Miller. Quelle: [W3C12]

Rolle. [BZ06]

2.4 SOAP

SOAP (*Simple Open Access Protocol*) bezeichnet ein Netzwerkprotokoll zum Austausch XML-basierter Daten zwischen Applikationen über HTTP und liegt aktuell in der Version 1.2 als W3C Empfehlung vor. Es ist plattform, betriebssystem und programmiersprachenunabhängig. [Sch10a]

2.4.1 Aufbau einer SOAP Nachricht

Eine SOAP Nachricht ist ein XML Dokument, bestehend aus folgenden Elementen [Sch10a] :

- Envelope, der es als SOAP Message identifiziert
- Header als optionales Element zum Übertragen von Systeminformationen
- Body mit Aufruf-und Antwortinformationen
- Fault Element, das Fehler und Statusinformationen beinhaltet.

Eine SOAP Nachricht muss verpflichtend den Envelope enthalten. Der Encoding Namespace sowie ein oder mehrere Attachements sind optional. 2.3 Unter Encoding Namespace ist das verwendete Kodierungsschema für die Nachricht zu verstehen. Wenn dieses nicht explizit angegeben wird, wird ein Standardschema verwendet. [Qua12]

Der Vorteil der Flexibilität durch die Codierung mittels XML wird durch Performancenachteile erkauft. Die Codierung/Decodierung der XML-Nachricht beim Versand und Empfang sowie die notwendige Schemavalidation führen zu erhöhtem Aufwand im Vergleich zu proprietären Protokollen in stark gekoppelten Systemen. [Mue01]

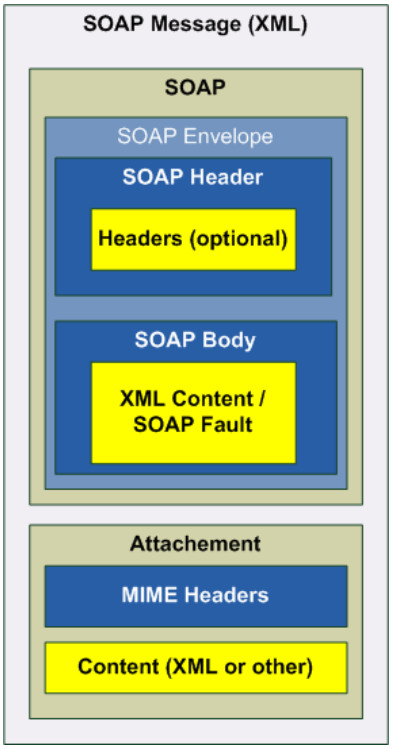


Abbildung 2.3: SOAP Nachricht - vgl. [Sch10a]

2.5 REST

REST (*Representational State Transfer Architecture*) bezeichnet ein Architekturprinzip von Webservices aufbauend auf den Gestaltungsprinzipien des World Wide Web. [Bay02]

Die Grundprinzipien von REST sind im folgenden Absatz kurz zusammengefasst [Til09]:

- Sämtliche Elemente des Web wie z.B. Webseiten, Bilder, Servlets, Skripts usw. stellen Ressourcen dar und sollen über eigene URIs mittels http angesprochen werden. Eine Ressource kann über mehrere Repräsentationen (Darstellungen in einem Standardformat) verfügen. Jede Ressource kann über Links in ihren Repräsentationen wieder auf andere Ressourcen verweisen. Dies ermöglicht es, einem Client Link zu folgen und so von einer Ressource zur nächsten zu wechseln.
- Die Kommunikation wird vom Client aus gesteuert, der aktiv Ressourcen vom passiven Server anfordert. Sie erfolgt über ein einheitliches Interface, die von sämtlichen Ressourcen unterstützt werden muss und mit den HTTP-Methoden GET, POST, PUT und DELETE angeboten wird. [Bay02]
- Der Status des Clients ist auf der Serverseite nicht bekannt. Sämtliche Anfragen an den Server müssen alle notwendigen Informationen beinhalten.

Vorteile [Bay02]:

- einfache Skalierbarkeit, da aufgrund des statuslosen Protokolls bei einem Wechsel von einem Server zum anderen kein Status mitgegeben werden muss.
- Lose Kopplung.

Nachteile [Cla03]:

- Definition von Ressourcen wesentlich: müssen exakt und richtig definiert werden.
- optionale Verflechtung mit dem HTTP-Protokoll problematisch, da alle Ressourcen über HTTP und URIs abgebildet werden.
- Abbildung von Resssourcen ohne Repräsentation nicht möglich (z.B. Namespaces).

2.6 WSDL

WSDL (*Web Service Definition Language*) ist ein Standard der W3C und liegt aktuell in der Version 2.0 vor. Es handelt sich dabei um eine XML-basierte Metasprache zur Definition von Webservices und deren Endpunkte. WSDL ist unabhängig von Betriebssystemen, Programmiersprachen oder Protokollen. [Sch10c]

Es können folgende Hauptelemente zum Einsatz kommen [Sch10c]:

- **<types>** Beschreibt die zum Austausch der Nachrichten verwendeten Datentypen in XML-Schema Syntax
- **<message>** Beschreibt den Aufbau der ausgetauschten Nachricht (Abfolge, verwendete Datentypen), wobei jede Nachricht aus mehreren Teilen bestehen kann. Sie ist vergleichbar mit Parametern in einem klassischen Funktionsaufruf.
- **<portType>** An dieser Stelle wird das Webservice selbst beschrieben. Die zur Verfügung stehenden Methoden sowie die verwendeten Nachrichten werden definiert.
- **<binding>** Beschreibt das verwendete Nachrichtenformat und die für das Webservice eingesetzten Protokolldetails.

WSDL wird zumeist in Kombination mit XML und SOAP für im Internet angebotene Webservices eingesetzt. Die WSDL-Spezifikation ermöglicht es dem Client zu erkennen, welche Funktionen mit welchen Datentypen zur Verfügung stehen. Durch den Einsatz von Codegeneratoren kann der benötigte Source Code automatisch aus der WSDL-Datei erzeugt werden.

2.7 OpenSearch

Bei *OpenSearch* handelt es sich um einen XML basierten Standard zum Austausch von Suchanfragen und Suchergebnissen zwischen unterschiedlichen Suchmaschinen. [Ope11] Er liegt aktuell in der Version 1.1 vor und wurde ursprünglich von Amazon[1] bzw. A9[2] entwickelt und unter der Creative Commons Attributions-ShareAlike v2.5 Lizenz lizensiert. [M.F08] Laut Mayer-Föll besteht OpenSearch aus drei Hauptbestandteilen [M.F08]:

- einer Beschreibungsdatei des Suchinterfaces oder der Webseite, die den Zugriff auf die Suchmaschine ermöglicht.
- einem Ausgabeformat für Suchergebnisse (*OpenSearch-RSS*). Suchmaschinen haben die Möglichkeit hier zusätzliche Metadaten, die von *RSS-Readern* nicht dargestellt werden können, hinzuzufügen. [Ope11]
- speziellen Feedreadern, die OpenSearch Suchergebnisse interpretieren und darstellen können.

OpenSearch ermöglicht es somit, je nach Art der Suchanfrage unterschiedliche Suchmaschinen einzusetzen, die miteinander kommunizieren und deren Suchergebnisse auf einfache Weise zusammengefügt werden können. [Ope11]

Die folgenden Sourcecodebeispiele, entnommen aus der OpenSearch Spezifikation [Ope11] zeigen ein Beispiel einer einfachen Beschreibungsdatei, einer Suchanfrage, sowie einer OpenSearch-RSS Datei.

Beispiel einer einfachen Beschreibungsdatei:

```
<?xml version="1.0" encoding="UTF-8"?>
 <OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
   <ShortName>Web Search</ShortName>
   <Description>Use Example.com to search the Web.</Description>
   <Tags>example web</Tags>
   <Contact>admin@example.com</Contact>
   <Url type="application/rss+xml"
        template="http://example.com/?q={searchTerms}&amp;pw={startPage?}&amp;
        format=rss"/>
 </OpenSearchDescription>
```

[1] www.amazon.com
[2] www.a9.com

Beispiel eines Suchanfrage-Elementes mit erweiterten Abfrage-Attributen:

```
<Query xmlns:custom="http://example.com/opensearchextensions/1.0/"
  role="example"
  searchTerms="cat"
  custom:color="blue"
  title="Sample search" />
```

Beispiel eines Suchergebnisses in *RSS 2.0* Format

```
<?xml version="1.0" encoding="UTF-8"?>
 <rss version="2.0"
       xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
       xmlns:atom="http://www.w3.org/2005/Atom">
   <channel>
     <title>Example.com Search: New York history</title>
     <link>http://example.com/New+York+history</link>
     <description>Search results for "New York history" at Example.com</description>
     <opensearch:totalResults>4230000</opensearch:totalResults>
     <opensearch:startIndex>21</opensearch:startIndex>
     <opensearch:itemsPerPage>10</opensearch:itemsPerPage>
     <atom:link rel="search" type="application/opensearchdescription+xml"
                href="http://example.com/opensearchdescription.xml"/>
     <opensearch:Query role="request" searchTerms="New York History" startPage="1" />
     <item>
       <title>New York History</title>
       <link>http://www.columbia.edu/cu/lweb/eguids/amerihist/nyc.html</link>
       <description>
         ... Harlem.NYC - A virtual tour and information on
          businesses ...  with historic photos of Colombia's own New York...
          ..... neighbourhood...
       </description>
     </item>
   </channel>
 </rss>
```

Kapitel 3

Verwandte Arbeiten

Das Konzept einer digitalen Bibliothek hat sich im Zuge der technischen Fortschritte in den letzen Jahren ständig geändert und weiterentwickelt. Was genau ist nun unter einer digitalen Bibliothek oder einem DLMS zu verstehen und welche Digitalen Bibliotheksmanagementsysteme existieren? Mit diesem Thema hat sich auch das DELOS Netzwerk (*Network of Excellence on Digital Libraries*), ein von der Europäischen Union unterstütztes Forschungsprojekt, beschäftigt. Als Ergebnis der gemeinsamen Anstrengungen wurde ein Manifest („*Digital Library Manifesto*“) herausgegeben, das sich mit den Eigenschaften und dem Aufbau digitaler Bibliotheken auseinandersetzt. Im folgenden Abschnitt finden sich einige Begriffsdefinitionen und Kernaussagen aus dem DELOS Manifest [CCI*07], Beispiele zu existenten DLMS Systemen sowie ein Vergleich dieser Systeme mit PROBADO.

3.1 Begriffsdefinitionen

Im Anschluss werden die Begriffe “*Digitale Bibliothek*”, *“Digitales Bibliothekssystem*” sowie *“Digitales Bibliotheks Management System*” näher erklärt. [CCI*07]

3.1.1 Digitale Bibliothek (DL)

Eine möglicherweise virtuelle Organisation zur lang andauernden umfassenden Sammlung, Verwaltung und Aufbewahrung von reichhaltigen digitalen Inhalten, die ihren Benutzergemeinschaften spezialisierte Funktionalitäten von messbarer Qualtiät und nach festgeschriebenen Richtlinien anbietet. [CCI*07]

3.1.2 Digitales Bibliothekssystem (DLS)

Ein Software System basierend auf einer (unter Umständen verteilten) Architektur, das jedwede Funktionalität anbietet, die von einer bestimmten digitalen Bibliothek benötigt wird. Benutzer interagieren mit einer digitalen Bibliothek über das zugehörige digitale Bibliothekssystem. [CCI*07]

3.1.3 Digitales Bibliotheks Management System (DLMS)

Eine generische Software, welche die entsprechende Software Infrastruktur für die folgenden Aufgaben anbietet:[CCI*07]:

1. die Herstellung und Administration eines digitalen Bibliothekssystems, das den grundlegenden Funktionsumfang für digitale Bibliotheken beinhaltet und
2. die Integration zusätzlicher Software für verfeinerte, spezialisierte oder erweiterte Funktionalität.

Ein DLMS ist also in die Kategorie „Systemsoftware“ einzuordnen und lässt sich in die folgenden Typen einteilen [CCI*07]:

- **Erweiterbares Digitales Bibliothekssystem**
  Ein voll funktionsfähiges DLMS mit einer definierten Kernfunktionalität und einer offenen Architektur, die es ermöglicht auf einfache Weise Zusatzkomponenten zu integrieren. DLS können mühelos durch Instanziierung des DLMS erzeugt werden.

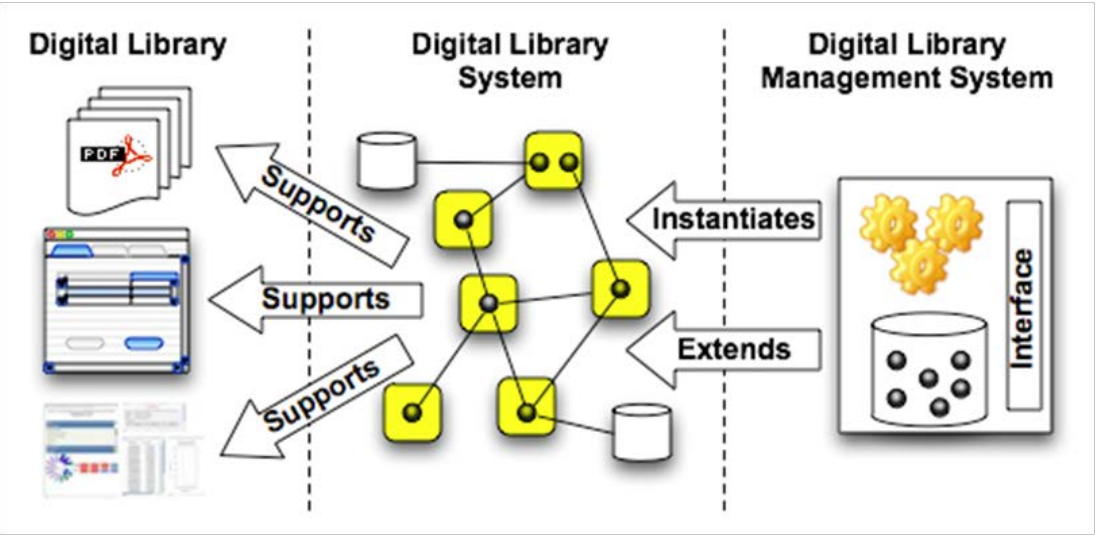


Abbildung 3.1: Framework in drei Schichten. Quelle: [CCI*07]

- **Digitales Bibliothekssystem Warenhaus**
  Eine Menge von Softwarekomponenten, welche die Kernfunktionalitäten einer digitalen Bibliothek abbilden, sowie eine Anzahl von Tools, um diese zu kombinieren, um DLS mit maßgeschneiderten Funktionalitäten zu erzeugen. Die Kombination neuer Softwarekomponenten mit bestehenden ist auf einfache Weise möglich.
- **Digitaler Bibliothekssystem Generator**
  Ein hochparametriertes Softwaresystem das Templates kapselt, die einen umfassenden Funktionsbereich beinhalten, der sowohl die Kernfunktionalitäten eines DLS als auch erweiterte Funktionalitäten beinhaltet, die für einen bestimmten Einsatzbereich als adäquat erachtet werden. Nach einer initialen Parameterkonfiguration, wird automatisch ein DLS bereit für Installation und Deployment generiert.

3.2 Ein Framework in drei Schichten

Aus diesen Definitionen lässt sich ein dreischichtiges Framework ableiten, das durch eine Reihe von Entwicklungsschritten zur Entstehung einer digitalen Bibliothek führt. Die Abbildung 3.1 entnommen aus dem Delos Referenzmodell [CCI*07] gibt einen Überblick über die Zusammenhänge.

3.3 Kernkonzepte zur Definition einer Digitalen Bibliothek

Nach Meinung der Autoren des DELOS Manifests gibt es, trotz der Vielzahl und der Verschiedenheit der existierenden digitalen Bibliothekssysteme, sechs Kernkonzepte, die allen gemeinsam sind. Wobei sich die ersten fünf (Inhalt, Benutzer, Funktionalität, Qualität und Richtlinien) auf digitale Bibliotheken und das letzte Konzept (Architektur) auf Managementsysteme für digitale Bibliotheken bezieht. Die Abbildung 3.2, entnommen aus dem DELOS Manifest [CCI*07], stellt dies graphisch dar.

3.3.1 Inhalt

Unter dem Inhalt einer digitalen Bibliothek sind die Daten und Informationen zu verstehen, die diese speichert (in Form von wie auch immer gearteten Informationsobjekten) und den Nutzern zur Verfügung stellt. Metadaten spielen in diesem Zusammenhang eine zentrale Rolle, da sie wertvolle Informationen zur syntaktischen, sematischen und kontextuellen Interpretation liefern. [CCI*07]

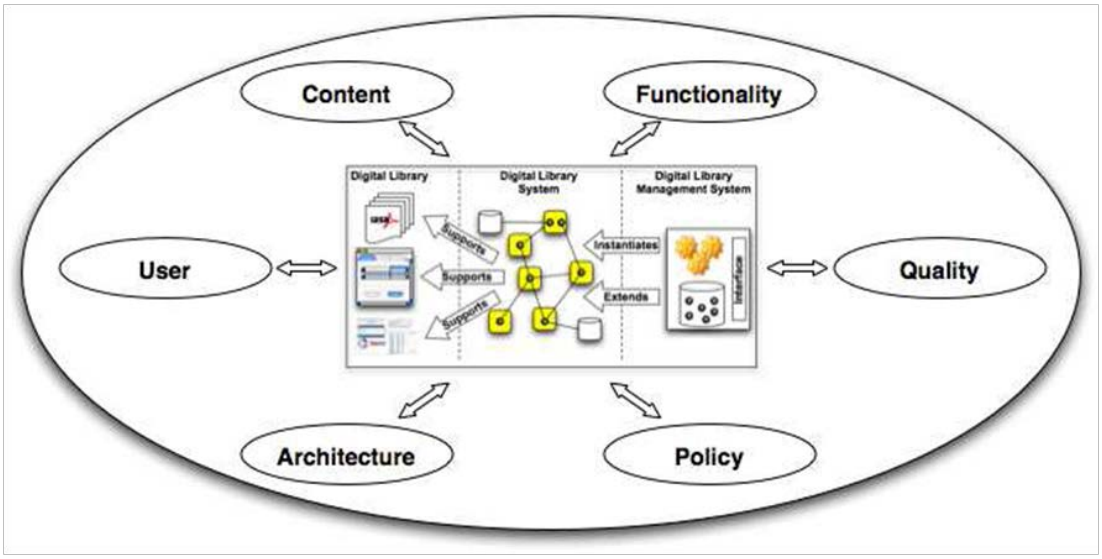


Abbildung 3.2: Das digitale Bibliotheksuniversum: Hauptkonzepte. Quelle: [CCI*07]

3.3.2 Benutzer

Dieses Konzept beschäftigt sich mit den Personen oder Applikationen die mit digitalen Bibliotheken interagieren und der Art und Weise, wie sie innerhalb der Bibliothek repräsentiert und verwaltet werden. Es umfasst Bereiche wie Benutzerprofile oder Rechte. Die Akteure lassen sich in vier Rollen einteilen [CCI*07]:

- **DL End-User**
  Die Gruppe der Einzelpersonen, die die digitale Bibliothek letztendlich benutzen.
- **DL Designer**
  Übernehmen die Aufgabe der applikationsseitigen Gestaltung der digitalen Bibliothek.
- **DL System Administrator**
  Benutzergruppe zur Administration der physikalischen Systeminfrastruktur
- **DL Applikationsentwickler**
  Wird benötigt zur Entwicklung von Softwarelösungen zur Realisierung einer digitalen Bibliothek.

Eine weitere wichtige Rolle, die in diesem Zusammenhang erwähnt werden sollte, ist die des DL Content Providers. Hier handelt es sich vor allem um Bibliotheken/Organisationen, die einerseits bestehende Sammlungen an digitalen Inhalten zur Verfügung stellen bzw. andererseits vorhandene Inhalte digitalisieren möchten. Im Rahmen des DELOS Manifests wird diese Rolle durch Teilaspekte der oben erwähnten Endbenutzer bzw. System Administrator Rollen abgedeckt. [CCI*07]

3.3.3 Funktionalität

Unter der Funktionalität einer digitalen Bibliothek ist die Anzahl der Dienste, die sie ihren Benutzern anbietet, zu verstehen. Dabei bildet das Speichern, die Suche und das Browsen von Objekten die Minimalfunktionalität, die durch spezielle, auf den jeweiligen Benutzerkreis zugeschnittene Spezialdienste erweitert werden kann. [CCI*07]

3.3.4 Qualität

Unter Qualität versteht man die Menge der Parameter, die zur objektiven Beurteilung des Inhalts und des Verhaltens einer digitalen Bibliothek verwendet werden kann. Sie kann sowohl funktions- als auch objektbezogen sein . Ein Teil der Parameter kann automatisch ausgewertet werden (z.B. Antwortzeiten), andere können nur durch Benutzerbefragungen ermittelt werden (z.B. Gestaltung des Benutzerinterfaces). [CCI*07]

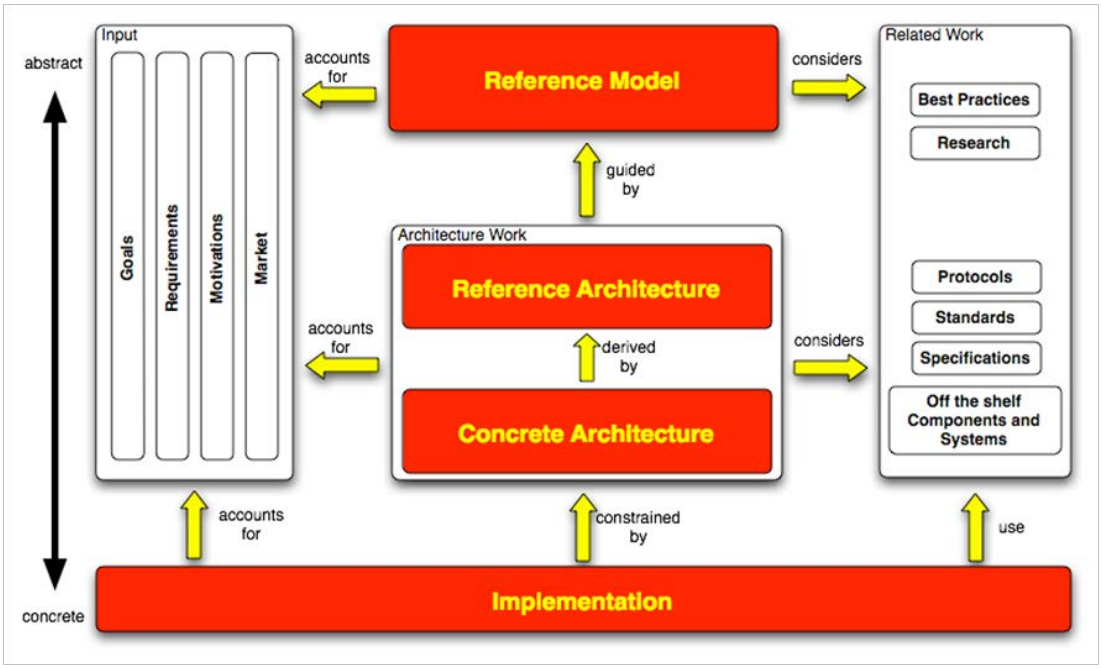


Abbildung 3.3: Zusammenhang zwischen den einzelnen Abstraktionsebenen. Quelle: [CCI*07]

3.3.5 Richtlinien

Richtlinien ist ein Sammelbegriff, der eine Menge von Bedingungen, Regeln und Begriffen umfasst, die die Interaktion zwischen Benutzern und der digitalen Bibliothek regeln. Dazu gehören unter anderem Richtlinien für das Benutzerverhalten, ein digitales Rechtemangement, Datenschutz und Kosten für Benutzer. [CCI*07]

3.3.6 Architektur

Als Architektur bezeichnet man die Hardware- und Softwarekomponenten, die eingesetzt werden, um die erwünschten Funktionalitäten und Daten abzubilden. Dabei wird zwischen drei Abstraktionsebenen unterschieden [CCI*07]:

- **DL Referenzmodell**
  Eine Menge von Konzepten, Axiomen und Beziehungen innerhalb einer spezifischen Wissensdomäne, die unabhängig von spezifischen Details, Technologien und konkreten Implementierungen sind
- **DLS Referenz Architektur**
  Ein architektonischer Design Pattern als abstrakte Lösung zur Implementierung der im Referenzmodell enthaltenen Konzepte. Hierbei gibt es (in Abhängigkeit vom Einsatzgebiet) unterschiedliche Referenzarchitekturen.
- **DLS Konkrete Architektur**
  Konkretisierung der Referenzarchitektur durch Einbindung von Standards und Spezifikationen, wie in Grafik 3.3 aus dem DELOS Manifest [CCI*07] dargestellt.

DLS Referenzarchitektur Dabei wird zwischen folgenden Architekturformen unterschieden [CCI*07]:

- **Service Oriented Architecture**
  Unter einer serviceorientierten Architektur (SOA) [ea10] versteht man eine flexible IT-Architektur, die Funktionen als von Anwendungen gemeinsam benutzbare Services anlegt. Zum Zugriff auf

diese Services wird von den Applikationen eine serviceorientierte Middleware Infrastructure eingesetzt. Da die Kommunikation über standardisierte Webservices erfolgt, gibt es keine Einschränkungen in Bezug auf Hersteller oder Plattform.

- **Peer-To-Peer Architecture (P2P)**
  Unter „*Peer-To-Peer Architecture*“ ist ein Netzwerk zu verstehen, in dem jeder Client über dieselben Fähigkeiten und Verantwortlichkeiten verfügt im Unterschied zur klassischen Client-Server Struktur.

- **Grid Computing**
  Mit dem Begriff „Grid Computing“ bezeichnet man die dynamische Kombination von geographisch verteilten autonomen Computer-Ressourcen aus verschiedenen Bereichen, die gemeinsam eine Art „*virtuellen Supercomputer*“ ergeben. Die verfügbaren Ressourcen werden in Abhängigkeit von der zu lösenden Aufgabe, der Kosten und der Performance ausgewählt und dynamisch kombiniert. [Min04]

Da die Anforderungen an eine digitale Bibliothek eine hoch skalierbare, flexible und anpassbare Infrastruktur erfordern, wird als Architektur eine Kombination der angeführten Konzepte und Techniken vorgeschlagen. [ABC*06]

3.3.7 DLS Zusammenhang zwischen den Kernkonzepten

Im folgenden Abschnitt soll der Zusammenhang zwischen den oben erläuterten Kernkonzepten näher aufgezeigt werden. Von den hier angeführten sechs Kernkonzepten sind drei (Architektur, Benutzer sowie Inhalt) unabhängig vom Konzept der digitalen Bibliothek. Funktionalität dient dazu, dem Benutzer den von ihm gewünschten Inhalt zur Verfügung zu stellen, wobei vorgegebene Richtlinien einzuhalten sind, um einen Qualitätsstandard sicherzustellen. Die Abbildungen 3.4, 3.5, entnommen aus dem DELOS Manifest, stellen diese Zusammenhänge bildlich dar. [CCI*07] Die Beziehungen zwischen den Hauptkonzepten einer digitalen Bibliothek sind im Zusammenhang mit der Art und Weise zu sehen, wie unterschiedliche Benutzergruppen mit dem System interagieren.

Dabei ist die Interaktion der einzelnen Benutzergruppen mit den verschiedenen Ebenen des digitalen Bibliotheksuniversums hierarchisch aufgebaut, wie in der Abbildung 3.6 aus dem DELOS Manifest [CCI*07] dargestellt.

End-User befinden sich an der Spitze der Pyramide, da sie nur mit der digitalen Bibliothek interagieren. Designer, System Administratoren und Anwendungsentwickler hingegen arbeiten unter Verwendung eines digitalen Bibliotheksmanagementsystems auch mit dem digitalen Bibliothekssystem. Die Knoten und Pfeile in den einzelnen Benutzerebenen, sollen die Interaktion zwischen den Mitgliedern einer Benutzergruppe, sowie deren Umgang mit den, in der jeweiligen Ebene zur Verfügung stehenden Systemen, symbolisieren. [CCI*07]

3.4 DL Ressourcenmodell

Das Referenzmodell beschäftigt sich mit den bereits erwähnten fünf Hauptkonzepten oder Ressourcen: Inhalt, Benutzer, Funktionalität, Richtlinien und Qualität, wobei im Folgenden näher auf das Ressourcenmodell und dabei im Speziellen auf den Inhalt (die sogenannte „Content Domain“) eingegangen wird. Eine Ressource ist somit das generellste aller digitalen Bibliothekskonzepte, das jeglichen Inhalt umfasst. Unter Instanzen von Ressourcen sind Informationsobjekte in jeglicher Form (z.B. Dokumente, Bilder, Multimedia Objekte usw.), sowie Architekturkomponenten, Funktionen, Richtlinien und Qualitätskriterien zu verstehen. Dies wird durch das Referenzmodell 3.7, entnommen aus dem DELOS Manifest [CCI*07], dargestellt.

Jede Ressource muss einen eindeutigen Identifier besitzen, sie sollte Metadaten aufweisen und kann annotiert sein. Zur genaueren Erklärung der in obiger Grafik aufgezeigten Zusammenhänge, finden sich im folgenden Abschnitt eine Reihe von Begriffsdefinitionen. [CCI*07]

3.4.1 Content Domain

Unter der Content Domain (Inhaltsdomäne) versteht man sämtliche Objekte, die in irgendeiner Form mit der in der digitalen Bibliothek abgelegten Information in Verbindung stehen. Sie wird am besten

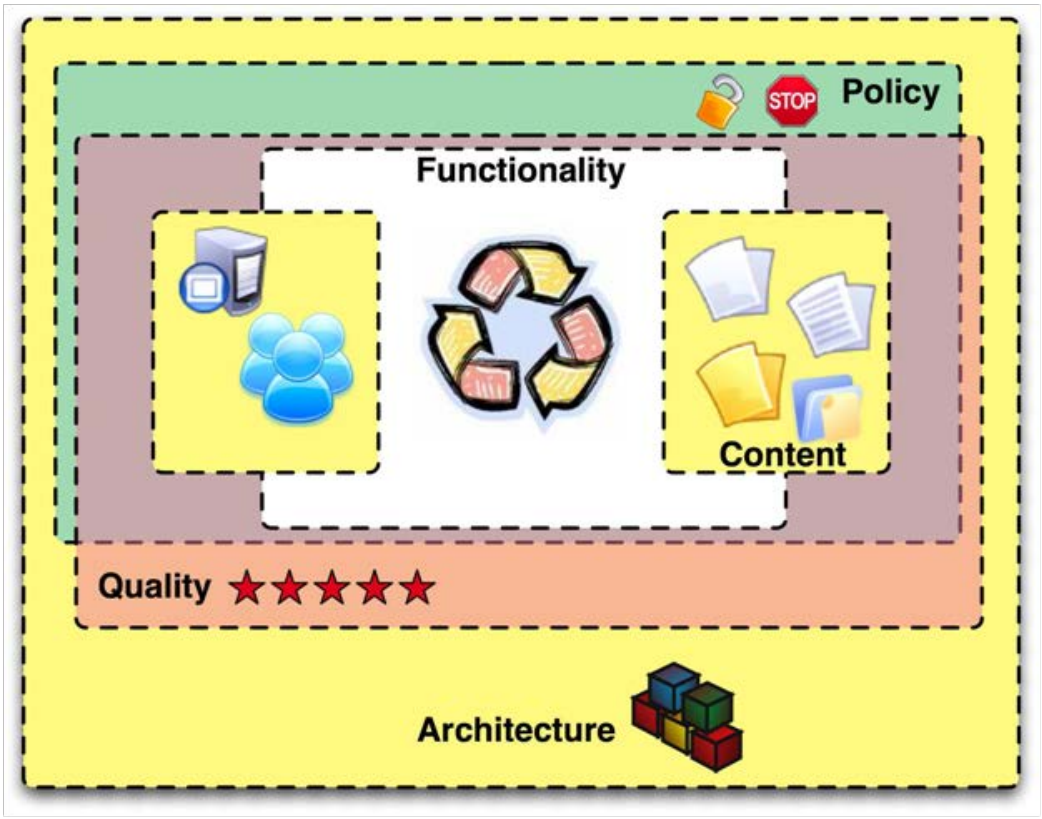


Abbildung 3.4: Beziehungen zwischen den Hauptkonzepten einer digitalen Bibliothek. Quelle: [CCI*07]

durch das folgende Konzept des Informationsobjektes beschrieben. [CCI*07]

3.4.2 Informationsobjekt

Die sogenannten „Informationsobjekte" repräsentieren beliebige Bibliotheksobjekte wie z.B. Bilder, Dokumente, 3D-Objekte, sowie Kombinationen dieser Objekte [CCI*07]. Dies wird in Abbildung 3.8, entnommen aus [CCI*07] bildlich dargestellt.

Informationsobjekte lassen sich anhand der Art und Weise in der sie zu anderen Informationsobjekten in Beziehung stehen in die folgenden Kategorien einteilen [CCI*07]:

- Primäre Informationsobjekte die für sich stehen (z.B. ein Bild oder ein 3D-Modell)
- Metadaten, deren Hauptaufgabe die nähere Beschreibung eines primären Informationsobjektes ist
- Annotationen, die Anmerkungen wie Notizen, Kommentare oder Links zu einem primären Informationsobjekt enthalten.

3.4.3 Metadaten

Wie bereits in Kapitel 2 erwähnt, lassen sich Metadaten in drei Kategorien unterteilen [Lib10]:

- Beschreibende Metadaten
- Administrative Metadaten
- Strukturelle Metadaten

Für den Benutzer eines Bibliothekssystems sind prinzipiell nur die beschreibenden Metadaten im Rahmen der Suche sichtbar. Administrative Metadaten werden innerhalb des Repository zur Verwaltung der

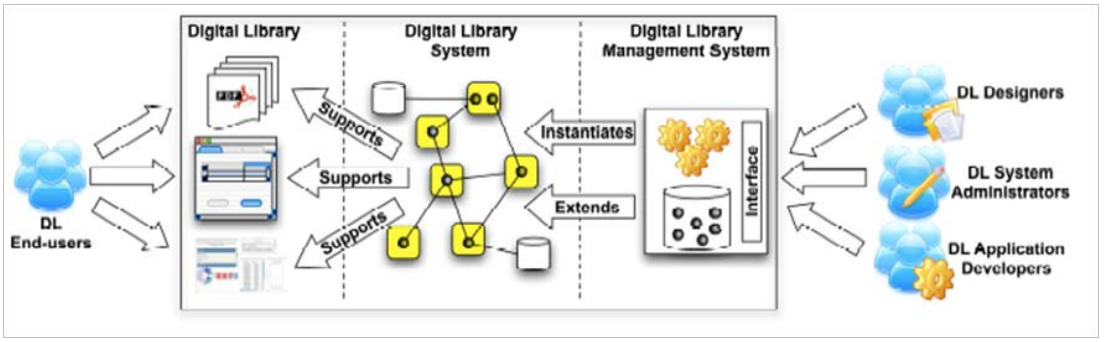


Abbildung 3.5: Rollen der Benutzer im 3-Schichtmodell. Quelle: [CCI*07]

Informationsobjekte verwendet. Strukturelle Daten werden im Rahmen der Suche vom Programminterface zur Suchoptimierung, Verknüpfung und Darstellung der Ergebnisse verwendet. [CCI*07]

3.5 Semantische Digitale Bibliotheken

In den vorangegangenen Kapiteln wurde versucht, näher auf die theoretischen Grundlagen und den Aufbau von digitalen Bibliotheken, einzugehen. Das aktuelle Kapitel, basierend auf dem Buch „*Semantic Digital Libraries*“, herausgegeben von Bill Mac Daniel und Sebastian Ryszard Kruk, versucht, den Begriff „*semantische digitale Bibliotheken*“ näher zu erläutern und die Unterschiede zur klassischen digitalen Bibliothek herauszuarbeiten. Die Suche in einer klassischen digitalen Bibliothek ist im Prinzip entweder im Sinne einer Metadatensuche oder einer Volltextsuche innerhalb des gespeicherten Objekts bzw. seiner Beschreibungen zu verstehen. Diese Art der Suche würde also Abfragen wie „Zeige alle 3D Objekte, die im Jahr 2008 entstanden sind“ oder „Zeige alle Symphonien, die von Johann Sebastian Bach komponiert wurden“, unterstützen. Eine Suchanfrage nach „allen Objekten, die in Bezug zum Thema Romantik stehen“, würde, vor allem wenn der Begriff Romantik nicht in der Datenbank vorkommt, keinen Treffer liefern. Dieses Beispiel zeigt die Bedeutung semantischer Daten zur Verbesserung von Suchergebnissen. Jene können entweder in Form von Metadaten zu den einzelnen Bibliotheksobjekten oder in Form von Ontologien abgelegt werden, um von speziellen Suchmaschinen mit semantischen Fähigkeiten ausgewertet zu werden. Ein Problem, das sich in diesem Zusammenhang stellt, ist die Abbildung semantischer Metadaten auf die zugrundeliegende Datenbankstruktur der digitalen Bibliothek. RDF Darstellungen von Metadaten und Ontologien sind von ihrer Struktur her azyklische Graphen und lassen sich nur schwer in ein relationales Datenbankschema pressen. Eine weitere Schwierigkeit liegt darin, dass ein Großteil dieser semantischen Zusammenhänge erst aus sprachlichen Beschreibungen, allgemeinen Erfahrungen, geschichtlichen Ereignissen etc., extrahiert werden muss. Hier spielen, zusätzlich zur Semantik, auch Ontologien eine wesentliche Rolle. Das durch sie vorgegebene Vokabular und Regelwerk ermöglichen es dem Suchalgorithmus erst, semantische Zusammenhänge zu berücksichtigen. Informationen müssen also analysiert und Beziehungen zwischen Konzepten, Begriffen und Ideen erkannt werden (Stichwort Wissensorganisationssystem). Um diese Zusammenhänge genauer betrachten zu können, werden im Folgenden einige der verwendeten Begriffe genauer erklärt. Im Anschluss daran wird kurz auf existierende semantische digitale Bibliothekssysteme eingegangen. [KM09]

3.5.1 Konzept

Ein Konzept ist nach Definition als Menge von Objekten, die mit Attributen verbunden sind, zu verstehen. [WB04] Dies kann am Beispiel eines Hauses 3.9 dargestellt werden:
Objekte und Attribute sind folgendermaßen miteinander verknüpft:

- Alle zu einem Konzept gehörigen Objekte weisen alle Attribute des Konzeptes auf. Alle Attribute werden von allen Objekten geteilt.
- Die Objekte werden als „*Extension*“ und die Attribute als „*Intension*“ bezeichnet.

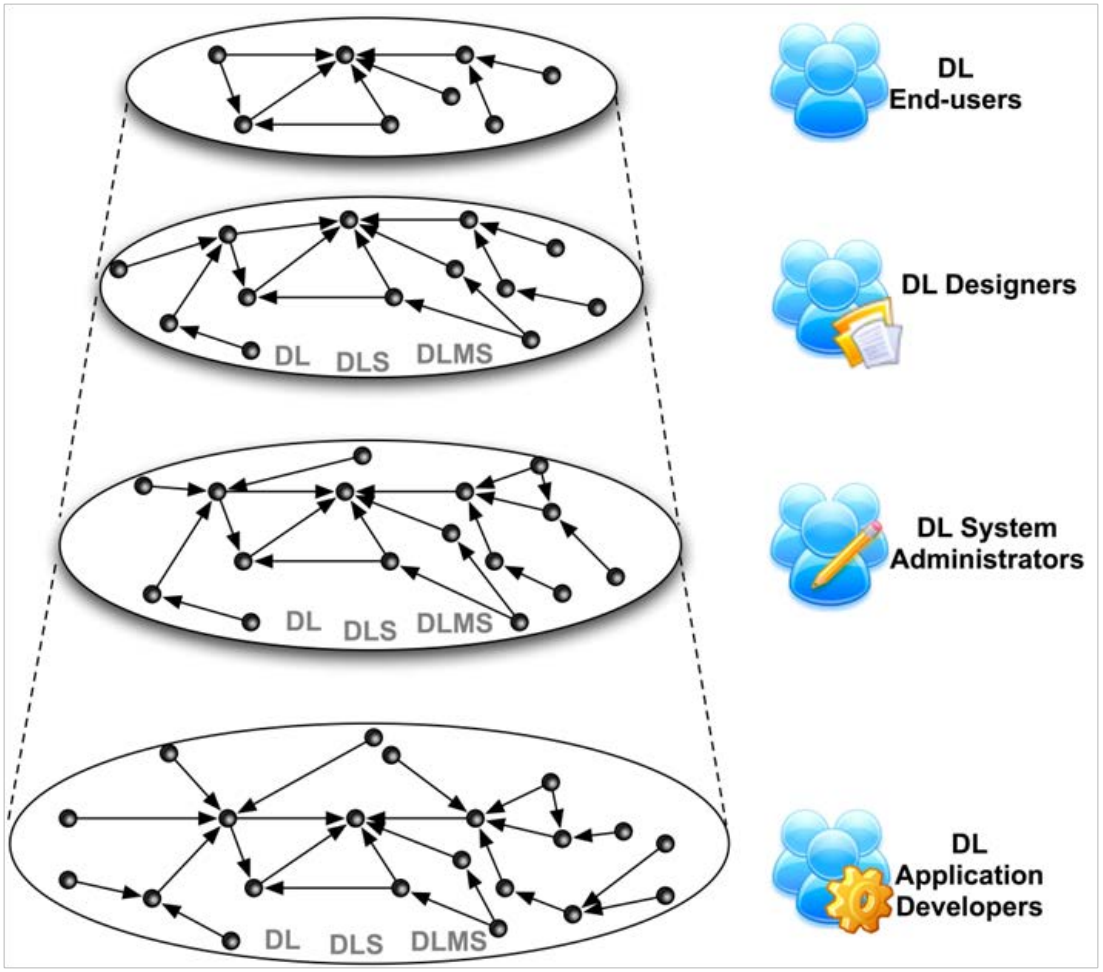


Abbildung 3.6: Interaktion von Benutzergruppen. Quelle: [CCI*07]

3.5.2 Knowledge Organisation System (KOS)

Die Hauptaufgabe eines Wissensorganisationssystems ist es, den Benutzer dahingehend zu unterstützen, das zu finden, was er eigentlich im Sinn hat. Nach der Definition von Dagobert Soergel besteht ein KOS aus zwei Ebenen und ist wie folgt aufgebaut [Soe09]:

1. Ebene: Beziehungen zwischen Konzepten und Begriffen („Concept-Term relationships“)
2. Ebene: Konzeptuelle Strukturen („conceptual structures“)
   - 2.1 Semantische Analyse und Facetten
   - 2.2 Hierarchie
   - 2.3 Interaktion zwischen Hierarchie und Facetten
   - 2.4 Verfeinerte Beziehungen zwischen Konzepten

Unter der ersten Ebene ist der Umgang mit Synonymen (alternative Begriffe für ein und dasselbe Konzept) und Homonymen (derselbe Begriff für unterschiedliche Konzepte) zu verstehen. Die zweite Ebene umfasst die Prinzipien der Facettenanalyse (analysiert unterschiedliche Aspekte eines Begriffs) und der Konzept-Hierarchie (Grobkonzept, Feinkonzept). Nicht alle KOS unterstützen beide Ebenen, vor allem, wenn es sich dabei um Ontologien handelt, wird die zweite Ebene betont, und die erste vernachlässigt.

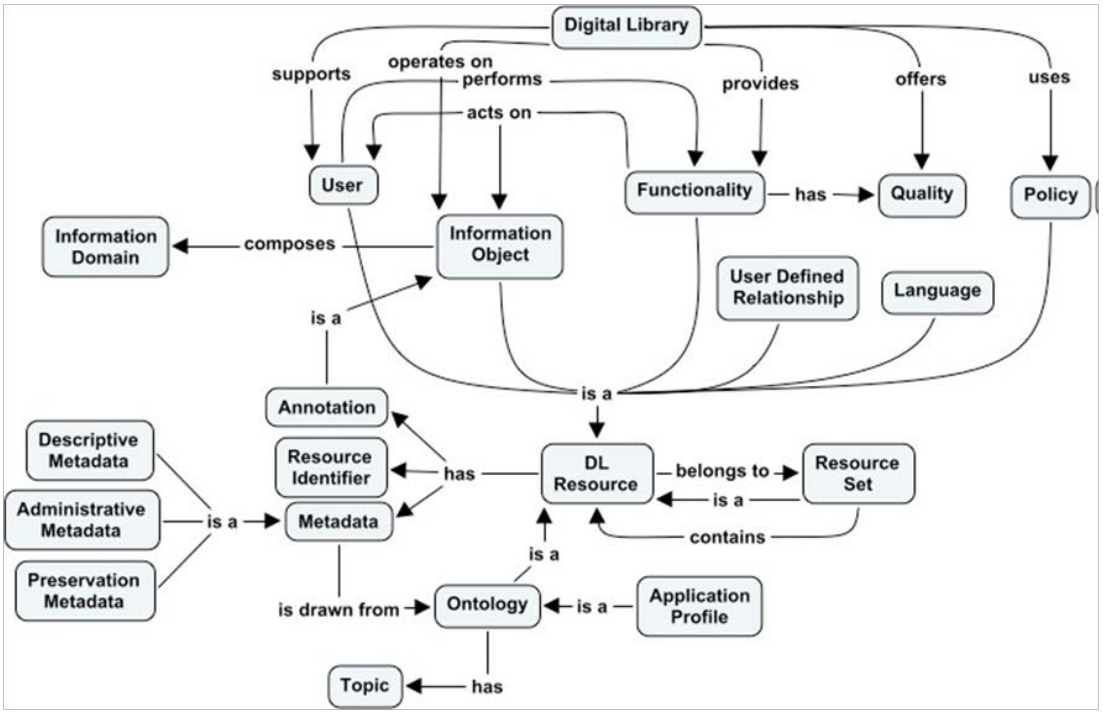


Abbildung 3.7: DL Referenzmodell (Concept Map zur Darstellung der Hauptkonzepte = Ressourcen. Quelle: [CCI*07])

3.5.3 Ontologie

Eine weitverbreitete Definition des Begriffes „*Ontologie*“ lautet: „*Eine Ontologie ist eine formale Spezifikation einer geteilten Konzeptualisierung*“. [Sem10] Im Grunde möchte man damit aussagen, dass unter einer Ontologie eine formale Beschreibung von Konzepten und deren Beziehungen zueinander, verstanden wird. Sie kann in weiterer Folge ‚dazu verwendet werden, aus vorhandenen Informationen, Schlussfolgerungen zu ziehen („reasoning“). [SDK09] Im Anschluss findet sich ein Beispiel für eine Ontologie, entnommen aus einem Projekt der Stanford University, die mittels OWL (Web Ontology Language) spezifiziert wurde [Sta10]:

```
<rdfs:Class rdf:ID="WINE">
   <rdfs:subClassOf rdf:resource="#POTABLE_LIQUID"/>
      <rdfs:subClassOf>
         <daml:Restriction>
            <daml:onProperty rdf:resource="#MAKER"/>
            <daml:minCardinality>1</daml:minCardinality>
         </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#MAKER"/>
            <daml:toClass rdf:resource="#WINERY"/>
        </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <daml:Restriction>
            <daml:onProperty rdf:resource="#GRAPE_SLOT"/>
            <daml:minCardinality>
               1
      ........
</rdfs:Class>
```

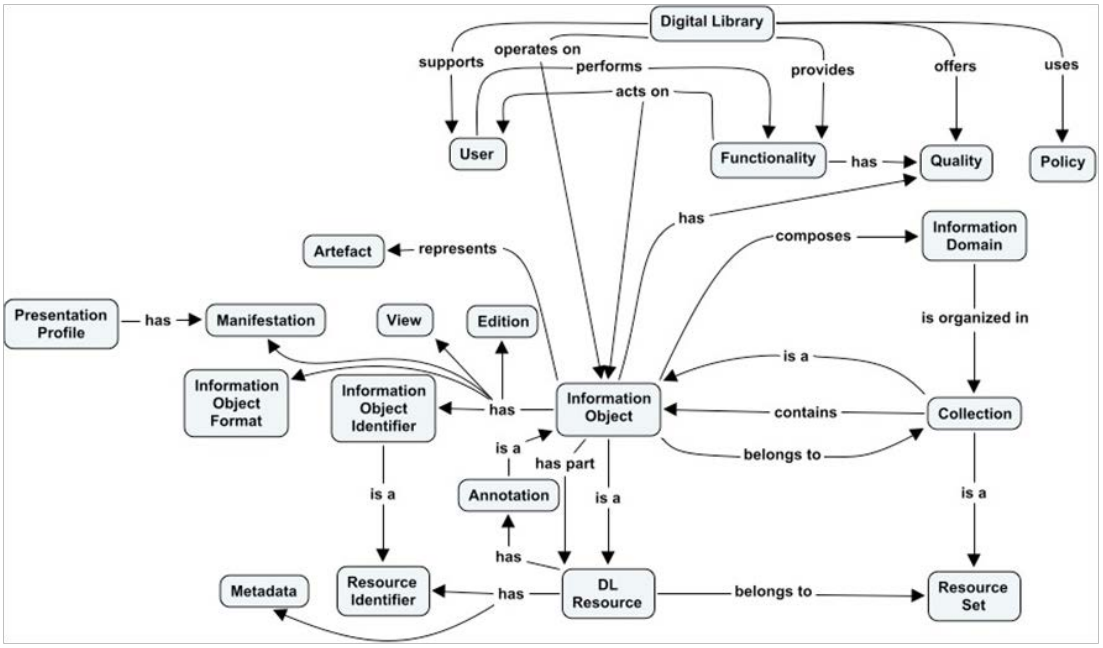


Abbildung 3.8: Definition eines Informationsobjektes. Quelle: [CCI*07]

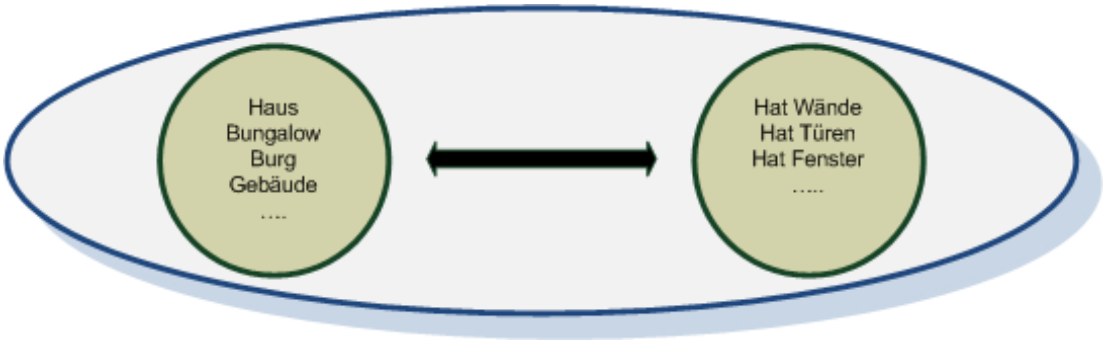


Abbildung 3.9: Objekte, Attribute und Beziehungen bilden ein Konzept. Quelle: [WB04]

Die hier dargestellte Ontologie definiert Wein als trinkbare Flüssigkeit, die von zumindest einem Weinhersteller produziert wird und aus mindestens einer Sorte Weintrauben hergestellt wird.

3.5.4 Soziale Semantische Informationsräume

Im Zuge der Weiterentwicklung des World Wide Web (Stichwort Web 3.0, Semantic Web) kommt es in zunehmendem Maße zur Entwicklung von sozialen, kollaborativen Netzwerken, in denen Benutzer Inhalte beisteuern, teilen, kommentieren und generell zusammenarbeiten. Facebook, Flickr, Wikipedia oder Bibsonomy wären hier als Beispiele zu nennen. [SDK09] In semantischen digitalen Bibliotheken sind Benutzer, im Gegensatz zu klassischen digitalen Bibliotheken, keine reinen Konsumenten von Inhalten, sie tragen auch Inhalte bei und zwar durch [KHP*06]:

- Tagging (Versehen von Objekten mit Stichwörtern) und Annotationen
- Folksonomies
- Soziales Semantisches Kollaboratives Filtern

Das Semantische Web ist also als Erweiterung des World Wide Webs zu verstehen, in dem Sinne, dass Inhalte in für Computer verständlicher Weise mit Bedeutung verknüpft werden, und somit besser ausgewertet werden können.

3.5.5 Ziele einer Semantischen Digitalen Bibliothek

Um die Charakteristika einer semantischen digitalen Bibliothek näher zu definieren, erwähnen Sebastian Ryszard Kruk und Bill McDaniel sieben im Rahmen des Delos-Projektes [ABC*06] erarbeite Ziele einer digitalen Bibliothek. [KM09]

1. Freier Zugang für alle
2. Alle gespeicherten Informationen sind zugänglich
3. Die Bibliothek steht jederzeit zur Verfügung
4. Ortsunabhängiger, personalisierter Zugriff (auch über mobile Geräte)
5. Benutzerfreundliches multimodales Benutzerinterface
6. Effizienter und effektiver Zugriff auf gespeicherte Objekte
7. Verteilte Architektur

Einer der wichtigsten Punkte hierbei ist der Zugriff auf Informationen. Um diesen effizienter gestalten zu können, spielt der Einsatz semantischer Technologien zur Erzeugung von Annotationen und Metadaten eine wesentliche Rolle. In diesem Zusammenhang wurden drei wesentliche Herausforderungen identifiziert [KM09]:

- Die Integration von Informationen aus verschiedenen Metadatenquellen (Benutzerprofile, Bookmarks, Beschreibungen usw.)
- Zusammenarbeit mit anderen Systemen durch den Einsatz von RDF
- Einsatz stabiler, flexibler, konfigurierbarer Benutzer- und Suchinterfaces mit semantischer Unterstützung (Verwendung von formalen und sozialen Annotationen), Benutzerprofilen, personalisierten Such- und Empfehlungsmechanismen und Community-Unterstützung

Um diesen Herausforderungen gerecht zu werden, sind einerseits die Schaffung eines dynamischen Objektmodells zur Speicherung komplexer Informationsobjekte und deren Beziehungen zueinander, andererseits die Implementierung einer Reihe als unverzichtbar anzusehender Dienste notwendig, die im Folgenden kurz aufgelistet sind [KM09]:

- Such-, Browsing- und Empfehlungsservice
- Annotationsmechanismen, sowohl automatisch als auch benutzerbasiert im Rahmen der Community
- Informationsverbreitungs- und Benachrichtigungsservices
- Sicherheits- und Zugriffsrichtlinien
- Kompatibilität, sowohl zu veralteten als auch zu modernen Protokollen
- Datensicherung und Archivierung
- Qualitätssicherung
- Benutzerdokumentation

3.5.6 Architekturkonzepte einer Semantischen Digitalen Bibliothek

Im folgenden Abschnitt, entnommen aus [KWK09], wird versucht, anhand der zuvor erarbeiten Anforderungen und Ziele, ein Architekturkonzept für eine semantische digitale Bibliothek zu erarbeiten. Grundsätzlich entsprechen Architektur und Aufbau einer semantischen digitalen Bibliothek dem einer klassischen digitalen Bibliothek. Es werden die selben Architekturkonzepte und Komponenten verwendet. Der Unterschied liegt im Verhalten der einzelnen Komponenten und in ihrer Interaktion zueinander. Der Benutzer einer semantischen digitalen Bibliothek versteht sich als Teil einer Gemeinschaft, die über

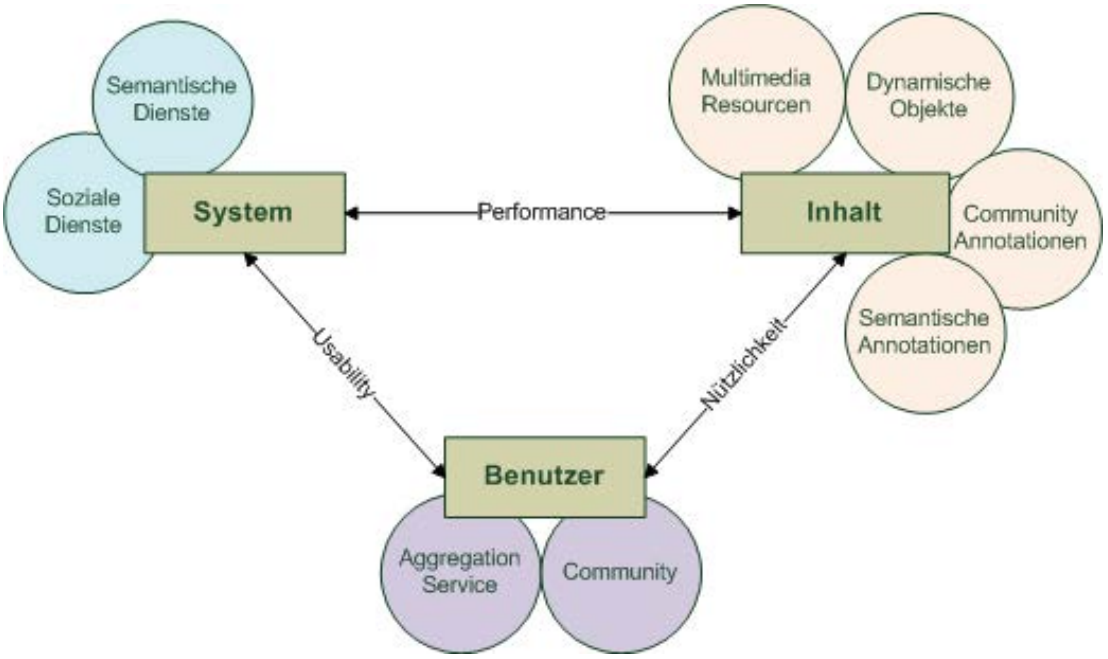


Abbildung 3.10: High Level Komponenten einer semantischen digitalen Bibliothek. Vgl. [KWK09]

die digitale Bibliothek miteinander kommuniziert. Um dies zu ermöglichen, muss das Bibliothekssystem um Dienste aus dem Bereich des Semantic Web und der sozialen Netzwerke erweitert werden. Die Grafik 3.10, nachempfunden einer Abbildung aus [KWK09], soll diese Zusammenhänge zum Ausdruck bringen.

Dies führt in weiterer Folge zu einer Schichtarchitektur, wobei sich die tieferen Schichten eher auf die gespeicherten Daten konzentrieren, während die höheren Schichten externe Dienste zur Verfügung stellen und mit dem Endbenutzer kommunizieren. Die Abbildung 3.11, gestaltet nach einer Vorlage aus, [KWK09] stellt diese Schichtarchitektur im Detail dar.

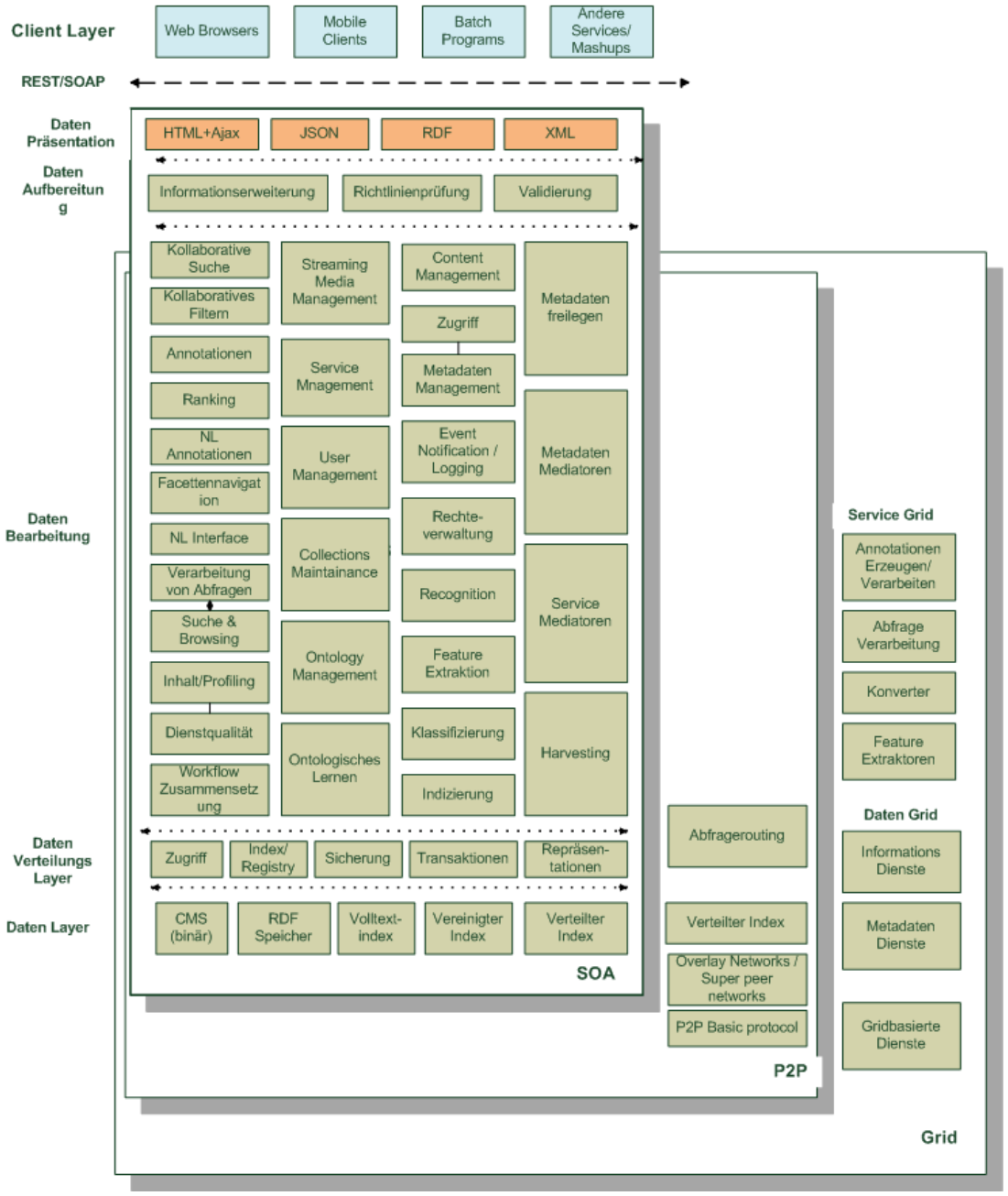


Abbildung 3.11: Schichtarchitektur einer semantischen digitalen Bibliothek. Vgl. [KWK09]

Kapitel 4

Beispiele für DLMS

Im folgenden Kapitel finden sich kurze Beschreibungen einiger bekannter Digital Library Management Systeme, sowie ein Vergleich der Eigenschaften dieser Systeme mit PROBADO.

4.1 DELOS DLMS

Mit dem *DELOS-DLMS* schuf die Delos Gruppe einen Prototypen eines digitalen Bibliothekssystems. Es besteht aus einer Anzahl spezialisierter digitaler Bibliotheks Services (ISIS), die in die OSIRIS Middleware (Open Service Infrastructure for Reliable and Integrated Process Support) integriert wurden. Die Kernstücke von ISIS und OSIRIS wurden an der ETH Zürich entwickelt. Es existiert sowohl eine in C++ entwickelte Version für Windows als auch eine in Java geschriebene Version für andere Plattformen. Besondere Merkmale von OSIRIS sind die Skalierbarkeit von Prozessen, sowie Verlässlichkeit und Transaktionalität. Die Serviceintegration erfolgt über WSDL (Beschreibung der Schnittstellen) und SOAP zur Kommunikation. Das System unterstützt inhaltsbasierte Abfragen auf Bilder, Audio, Video und 3D Modelle sowie traditionelle Abfragen mit Stichwörtern. Die Suche erfolgt über ein graphisches Userinterface, das auch Browsing erlaubt. Die erhaltenen Suchergebnisse können annotiert werden. Ein sprachgesteuertes Interface, sowie der Zugriff über „interactive paper“ stehen zur Verfügung. Weiters ist der Einsatz in einem verteilten Peer-To-Peer Netzwerk möglich. [BRS*09] Die Grafik 4.1 gibt einen Überblick über die Struktur von Delos DLMS. [BBC*]

4.2 BRICKS

BRICKS (*Building Resources for Integrated Cultural Knowledge Services*) war ein in den Jahren 2004-2007 durchgeführtes Projekt mit dem Ziel Open Source Software Lösungen zur Verfügung zu stellen, die es im Bereich der Pflege und Verbreitung des Weltkulturerbes tätigen Institutionen und Benutzern erlauben, Inhalte und Dienste miteinander zu teilen. Dabei wurde vor allem auf Eigenschaften wie Erweiterbarkeit, Skalierbarkeit, Verfügbarkeit, Kompatibilität Wert gelegt. Weiters sollte den teilnehmenden Institutionen ein breiter Bereich an Lösungen abhängig von der Art und Weise ihres Engagements angeboten werden können. Dabei sollte die Mitgliedschaft in der Bricks Community flexibel, einfach und mit geringen Kosten verbunden sein. Zur Umsetzung wurde eine dezentralisierte Peer-to-Peer Architektur gewählt, in der jeder Teilnehmer einen Knoten darstellt. Dabei sind jedem Knoten eine Anzahl seiner Nachbarn bekannt. Anfragen müssen an diese weitergeleitet werden, die sie wiederum weiterleiten. [Pro04] Die Grafik 4.2, entnommen aus der Online Edition von ERCIM News [MR05] zeigt die Architektur eines solchen Knotens.

Die gezeigten Komponenten sind Webservices, die sich grob in folgende Kategorien einteilen lassen [MR05]:

- Grundlegende Komponenten, deren Vorhandensein für die Funktion des Systems essentiell ist.
- Kernkomponenten wie Benutzermanagement, Rechteverwaltung und Suche
- Basiskomponenten enthalten optionale Komponenten wie z.B. Contentmanagement, Annotationsservices oder Metadatenverwaltung.

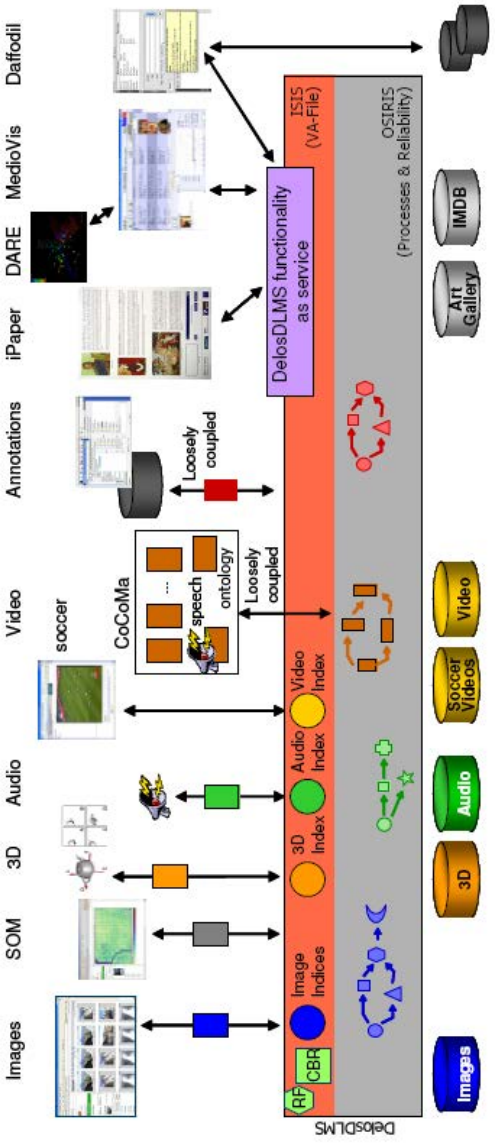


Abbildung 4.1: Überblick über Delos DLMS. Quelle: [BBC*]

Dabei wurden Kernapplikationen für folgende Bereiche vorgesehen [MR05]:

- Archäologie, die unter anderem die Bereiche Rekonstruktionen von Kulturlandschaften und Zugang zu visuellen Artefakten beinhaltet.
- Living Memory, das der Allgemeinheit ermöglichen sollte, eigene Inhalte beizutragen
- Scriptorium, das sich mit dem Austausch und der Verwaltung digitaler Texte und historischer Dokumente beschäftigt
- Europäisches Museumsforum, ein digitaler Bewerbungsprozess für den europäischen Museumspreis des Jahres.

4.3 DILIGENT

Unter dem *DILIGENT Framework* versteht man ein von der Europäischen Union gefördertes Projekt, das den Versuch unternimmt sogenannte „Distributed Information Retrieval“ Systeme (DIR) mit digi-

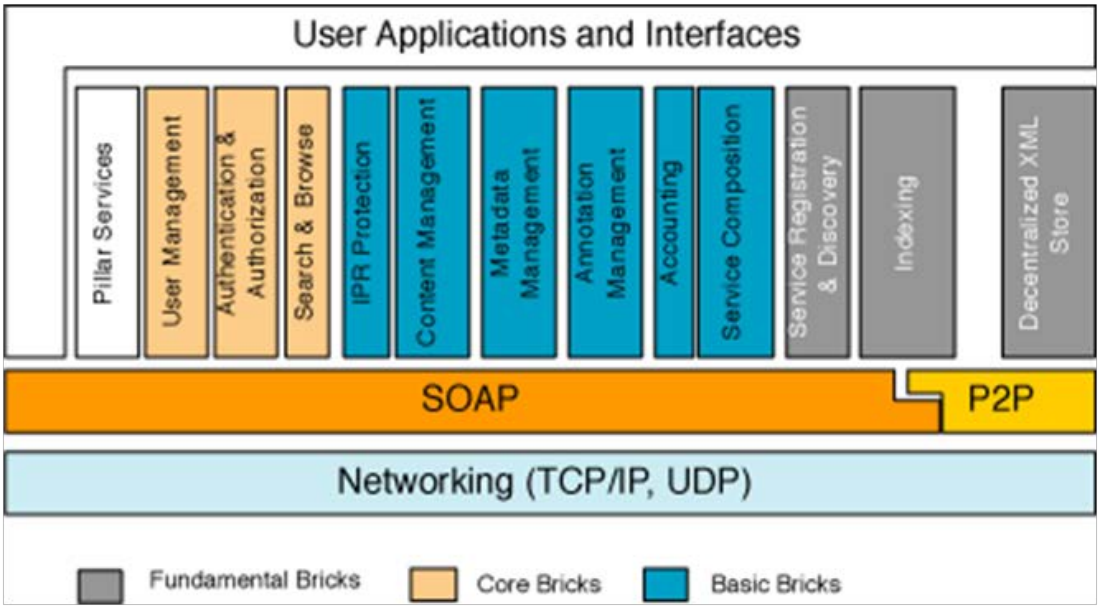


Abbildung 4.2: Komponenten eines BRICKS Knoten. Quelle: [MR05]

talen Bibliotheken zu kombinieren. Dies erfolgt durch das zur Verfügung stellen einer verteilten Infrastruktur, sowie einer Reihe von spezialisierten Anwendungen und Diensten für digitale Bibliotheken im Rahmen einer europäischen GRID Plattform. Dadurch soll die Gestaltung virtueller digitaler Bibliotheken ermöglicht werden, die einerseits von der Gemeinschaft zur Verfügung gestellte Daten beinhalten, andererseits nach Bedarf rechnerübergreifend eingesetzt werden können. Dabei unterstützt das DIR Framework (g-Cube) innerhalb der Projektinfrastruktur folgende Anwendungsdienste [SCB07]:

- Inhaltsselektion (Content Source Selection, CSS)
- Inhaltsbeschreibung (Content Source Dscription, CSD)
- Datenverschmelzung (Data Fusion, DF)

4.4 MARIAN

Bei MARIAN (*Multiple Access Retrieval of library Information with Annotations*) handelt es sich ursprünglich um ein von Virginia Tech entwickeltes Online Informationssystem für Bibliothekskataloge. Das System [GFF01] sollte dem Benutzer folgende Features zur Verfügung stellen [Tec10a]:

- Literatursuche nach Autor, Subject und Publikationsdatum
- Stichwortsuche nach verschiedenen MARC-Textfeldern u.a. Titel und Subject
- Autoren, Titel und Stichwörter in der Detailanzeige sind mit Hyperlinks versehen und lösen durch Anklicken eine weitere Abfrage aus. Das Anklicken des Autors führt also beispielsweise zur Anzeige aller Bücher dieses Autors.
- Annotationsmöglichkeit von Büchern durch den Benutzer

Das System ist modular aufgebaut und besteht aus folgenden Schichten

- **Interface Management Layer:** bildet die Schnittstelle zum Benutzer und enthält verschiedene Interface Manager sowie einen WWW-Gateway.
- **Session Management Layer:** enthält neben einem Session Manager Objekt auch Parser, Formatter, sowie Dienste für das Handling von Abfragen sowie Relevanz-Feedback.

- **Access Method Layer:** enthält verschiedene Dienste zur Ausführung und Kombination von Abfragen, sowie einen sogenannten Feedback Query Synthesizer zur Neugewichtung von Abfragen aufgrund von Benutzer Feedback.
- **Data Management Layer:** enthält die verwendeten Datenbanken wie Term, Text-Vektor, Link, MARC-Record Datenbanken sowie Datenbanken für die invertierten/indizierten Textkomponenten und Dokumente.

Das 1990 begonnene Projekt wurde zunächst in C++ entwickelt. Eine Java Reimplementierung begann 1999. [HKZZ99] Im Zuge der Reimplementierung wurde der Funktionsumfang des Systems zu einer globalen digitalen Bibliothek für Bücher, Dissertationen und Thesen - „Networked Digital Library for Dissertations and Thesis“ [Tec10b] erweitert. Besonderen Wert wurde dabei auf Skalierbarkeit, Performance und Flexibilität gelegt. [Fra01]

4.5 Fedora

Das Fedora (*Flexible Extensible Digital Repository Architecture*) Projekt, nicht zu verwechseln mit der gleichnamigen Linux-Distribution, wird von der Fedora Commons Community unter der Aufsicht der Non-Profit Organisation DuraSpace verwaltet. Es wurde ursprünglich an der Cornell University entwickelt und sollte als Framework zur Ablage und Verwaltung digitaler Objekte dienen. Es handelt sich dabei um ein leistungsfähiges, flexibles Open-Source Content-Managementsystem, das sowohl jegliche Art von digitalen Objekten als auch zugehörige Metadaten ablegen kann. Dabei kann die Ablage der Daten wahlweise in einer Datenbank oder im Dateisystem erfolgen. Der Zugriff auf das Repository kann wahlweise lokal als auch von extern über ein Webinterface (REST bzw. SOAP) erfolgen. Als Suchoptionen werden Volltextsuche sowie eine RDF Suche unter Verwendung von SPARQL angeboten. Im Falle eines Crashs steht ein Desaster-Recovery Mechanismus zum Wiederaufbau des Repository aus den enthaltenen Objekten zur Verfügung. Der Aufbau ist modular gestaltet, sodass dem Benutzer eine Anzahl verschiedener Frontends zur Verfügung stehen. Repository Events können über ein Java Message Service an interessierte Frontend Applikationen weitergeleitet werden. Fedora wurde in Java entwickelt und liegt aktuell in der Version 3.3 vor. Das Projekt wurde im Jahr 2001 ins Leben gerufen und wird laufend weiterentwickelt. Es wird von einer umfangreichen Community unterstützt und hat einen hohen Verbreitungsgrad. [DUR10]

4.6 REAP

Im Zusammenhand mit digitalen Bibliotheken und der Verbreitung von Inhalten über das Internet erlangt die Frage nach der Wahrung der Urheberrechte und in Folge dessen nach einem digitalen Rechtemanagement immer größere Bedeutung. Bei REAP (*Rights Enforcing Access Protocol*) handelt es sich um ein Zugriffsprotokoll, das innerhalb einer digitalen Bibliothek oder eines Repositorys zur Steuerung von Benutzerzugriffen auf die gespeicherten Inhalte verwendet werden kann. Die Vorgehensweise ist dabei wie folgt: Der Rechteinhaber legt initial fest, welche Zugriffsrechte ein Benutzer haben darf und welche Voraussetzungen dafür zu erfüllen sind („Offer“). Der Benutzer registriert sich im System. Dadurch wird ein Benutzerprofil angelegt und er hat die Möglichkeit ein sogenanntes „Agreement“ abzugeben, das ihm Zugriffsrechte auf den Inhalt der Bibliothek gewährt. Sowohl „Offers“ als auch „Agreements“ sind in einer spezifischen Rechtesprache basierend auf ODRL abgelegt. [Ves04] Unter ODRL (*Open Digital Rights Language*) versteht man einen von der Open Digital Rights Initiative entwickelten XML-basierten Standard für digitales Rechtemangement. [ODR10] Im Zuge einer Anfrage wird die URL des angefragten Objektes sowie die Session-ID an das REAP-Service übergeben, das dann prüft, ob der anfragende Benutzer über ein entsprechendes „Agreement“ verfügt um auf die angefragten Daten auch zugreifen zu können. Der Vorteil in der Verwendung von ODRL liegt darin, dass Rechtebeschreibungen somit auf einfache Weise erstellt bzw. zwischen verschiedenen Systemen ausgetauscht werden können. Es handelt sich bei REAP um einen Prototyp, der durchaus noch Schwachstellen aufweist. Es fehlt z.B. die Möglichkeit einem Benutzer Zugriffsrechte wieder zu entziehen, falls sich sein Profil dahingehend ändert, dass die notwendigen Vorbedingungen nicht mehr erfüllt sind. [Ves04]

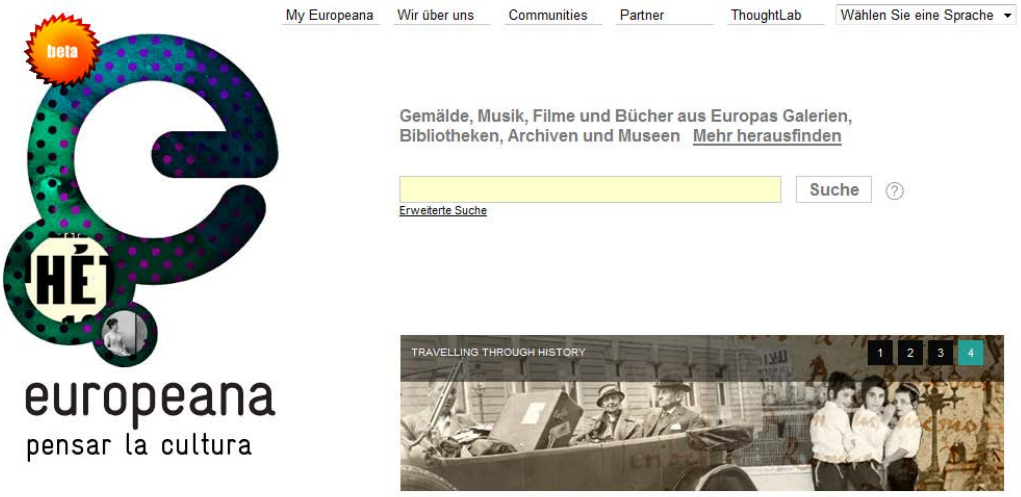


Abbildung 4.3: Europeana Such Interface. Quelle: [Eur11]

4.7 EUROPEANA

Bei *Europeana* handelt es sich um ein im Jahre 2007 begonnenes Projekt der Europäischen Kommission namens *eContentplus* mit dem Ziel, das kulturelle Erbe Europas der Öffentlichkeit zugänglich zu machen. Der Hauptsitz des Projektes ist die Nationalbibliothek der Niederlande. Neben der Europeana Projektgruppe leisten auch andere Projekte und Organisationen aus dem Bereich des Kulturerbes einen technologischen und inhaltlichen Beitrag und bilden gemeinsam mit der Projektgruppe die Europeana Gruppe. Ziel des Projektes ist es, Open-Source Lösungen für das Erschliessen von Dokumenten, Büchern, Museumsobjekten und audiovisuellen Dokumenten zur Verfügung zu stellen.

Im November 2008 wurde der erste Prototyp veröffentlicht. Aktuell umfasst die über Europeana zugängliche Sammlung über 10 Millionen Objekte. Seit März 2011 steht den Partnern von Europeana eine API (*Application Programming Interface*), sowie ein Search-Widget zur Verfügung, die es ermöglichen, auf möglichst einfache Weise über die eigene Webseite oder eine mobile Applikation Abfragen an Europeana zu stellen und Suchergebnisse bzw. Inhalte maßgeschneidert darzustellen. [Eur11] Die Europeana API basiert auf dem in Kapitel 2 beschiebenen *OpenSearch* Standard, einem auf XML basierenden Standard zum Austauch von Suchanfragen und Suchergebnissen zwischen verschiedenen Suchmaschinen. [Ope11] Wie in Abbildung 4.3 dargestellt, bietet Europeana eine einfache sowie eine erweiterte Suche unter Einsatz von Wildcards, Phrasen und logisch miteinander verknüpften Suchbegriffen. Das Suchinterface ist in verschiedenen Sprachen verfügbar. Die Suchergebnisse werden, nach Kategorien unterschieden, auf unterschiedlichen Aktenreitern dargestellt (siehe Abbildung 4.5). Der Benutzer hat die Möglichkeit, die Suchergebnisse über die Auswahl von Kategorien zu verfeinern bzw. sie in seinem persönlichem Konto abzuspeichern. 4.4

In den Jahren 2011-2015 sind, umschrieben mit den Schlagworten *“aggregate”*, *“distribute”*, *“facilitate”* und *“engage”*, Erweiterungen in folgenden Bereichen geplant:

- Erweiterung des Datenbestandes und des Netzwerkes der Kontributoren sowie die Verbesserung der Qualität der gewonnenen Metadaten.
- Schaffung eines Zuganges zu Inhalten für eine möglichst große Benutzergruppe.
- Stärkung des Kultursekors durch Wissensgewinnung und Innovation.
- Entwicklung neuer Möglichkeiten der Benutzerinteraktion.

Abbildung 4.4: Europeana Darstellung von Suchergebnissen. Quelle: [Eur11]

Abbildung 4.5: Europeana Verfeinerung von Suchergebnissen. Quelle: [Eur11]

4.8 PROBADO

(Prototypischer Betrieb Allgemeiner Dokumente) wurde im Februar 2006 als Kooperationsprojekt zwischen der Technischen Universität Darmstadt, der Universität Bonn, der Technischen Universität Graz, dem OFFIS Institut für Informationstechnologie in Oldenburg, der Technischen Informationsbibliothek Hannover sowie der Staatsbibliothek München ins Leben gerufen. Ziel des Projektes war es, einen Prototyp einer digitalen Bibliothek für nicht textuelle Objekte zu schaffen. Es sollte ein generisches Repository Framework geschaffen werden, das sich in der ersten Projektphase auf die Dokumententypen Musik, E-Learning-Materialien und 3D-Objekte konzentrierte. Dabei soll ein durchgängiger Ablauf, beginnend bei der Einpflege und Indizierung der Objekte, bis hin zur Suche und Ergebnispräsentation, unterstützt werden [KKSA07]. Der im Rahmen dieser Diplomarbeit mitentwickelte Prototyp wurde als Java-Webservice unter Verwendung einer MS-SQL Server Datenbank umgesetzt. Im Zuge der Projektumsetzung wurde der E-Learning Teil aus dem Projektumfang entfernt, sodass der fertige Prototyp sich auf die Spezialbereiche Musik und 3D beschränkt. Das System, siehe Abbildung 6.1, entnommen aus der ungekürzten Version der ECDL-Publikation aus dem Jahre 2010 [BBWS09], gliedert sich in drei Schichten:

- **Frontend Layer**
  Stellt die Ebene der Benutzerinterfaces dar. Dient zur Einpflege von Daten ins System, zur Ein-

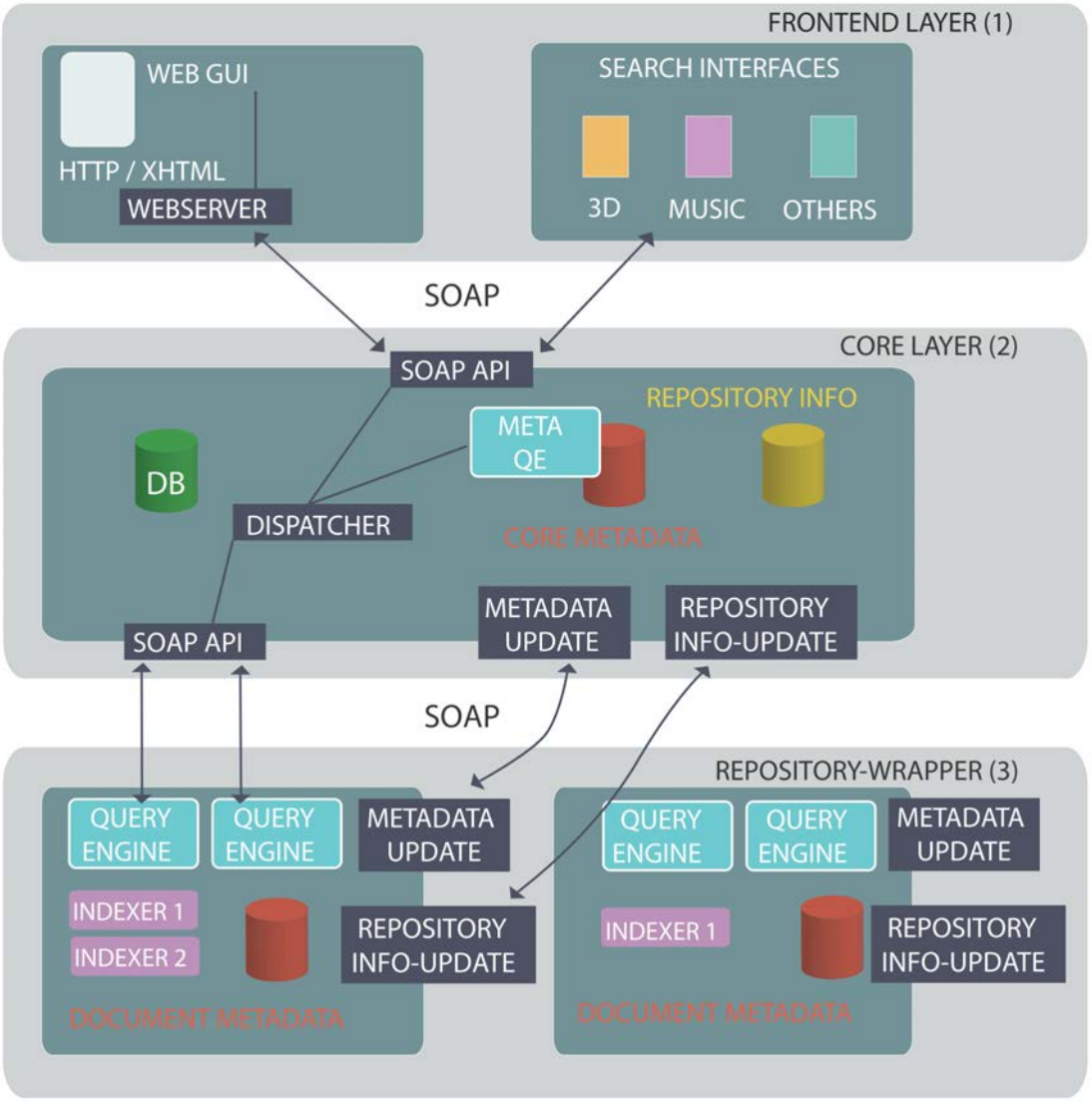


Abbildung 4.6: 3-Schichtarchitektur des PROBADO Systems. Quelle: [BBWS09]

gabe von Suchanfragen sowie zur Präsentation der Abfrageergebnisse. Kommuniziert über SOAP Anfragen mit dem Core Layer. Einfach erweiterbarer Webrahmen, der die Einbindung spezialisierter Interfaces für inhaltsbasierte Suchen ermöglicht. Diese Interfaces können entweder als domänenspezifische Browser Plugins in den Webrahmen eingebunden, oder aber auch als Standalone Clients zur Verfügung gestellt werden. [BKHS09]

- **Core Layer**
  Bildet die Middleware zur Verwaltung und Verteilung von Suchanfragen an die einzelnen Spezialrepositorien. Enthält einen Kerndatensatz an Metadaten (Teil des Dublin-CORE Schemas), der die in den Spezialrepositorien enthaltenen Objekte beschreibt, und im Zuge der CORE-Suche auch direkt mittels Volltext bzw. Metadatensuche durchsucht werden kann. Dieser Kerndatensatz wird über einen Updatemechanismus (Webservice) ständig aktuell gehalten. Weiters stehen eine Benutzer- und Rechteverwaltung sowie ein rudimentärer Feedbackmechanismus zur Verfügung. Anfragen werden zunächst an den Core-Layer gestellt und im Falle von Metadatenanfragen auch sofort beantwortet, im Falle inhaltsbasierter Abfragen an die Spezial-Repositorien weiter geleitet.

- **Repository Wrapper**
  Hier handelt es sich um die Ebene der über SOAP angebundenen Spezial Repositorien (3D, Musik). Diese zeichnen sich vor allem durch einen ausgefeilten Mechanismus der Datenvorbear-

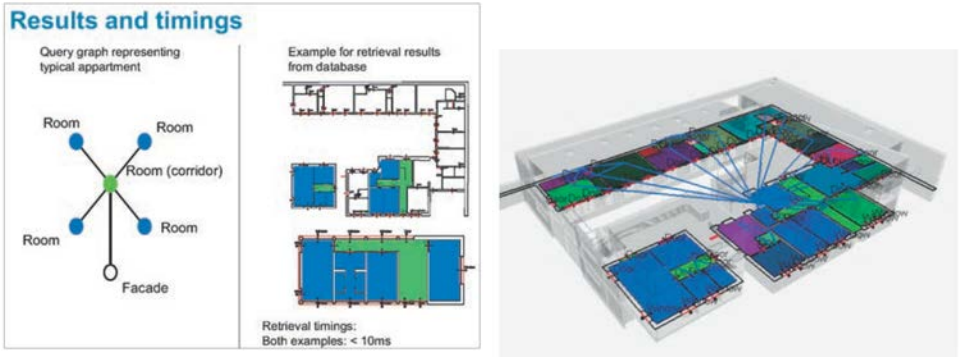


Abbildung 4.7: Suche über Raumverbindungsgraphen. Quelle: [SB09], [BBK*09]

beitung während der Einpflege, durch kontextbasierende Indizierung und Metadatengewinnung, sowie durch innovative Lösungen im Bereich der Abfrage (z.B. query-by-example) und Ergebnispräsentation, aus. [BBWS09]

4.8.1 Inhaltsbasierte Erschließung und Ergebnispräsentation

Die große Stärke von PROBADO liegt in der inhaltsbasierten Erschließung der gespeicherten Objekte. Bereits beim Einpflegevorgang werden über automatische Mechanismen technische Metadaten zu den gespeicherten Objekten gewonnen und Vorschaubilder generiert. Um die manuelle Katalogsarbeit nach Möglichkeit zu reduzieren, werden zusätzlich über die 3D Suchmaschine Indexdaten erstellt und abgelegt. Automatisch erkannte geometrische Charakteristika können mit bestimmten Klassen von manuell vorklassifizierten Objekten verglichen werden. [SB09] In diesem Kontext ist auch der Aufbau einer annotierten Datenbank für architekturspezifische Bauelemente zu sehen, in der geometrische Primitive zu semantischen Einheiten wie Gebäudeabschnitten, Stockwerken usw. zusammengefasst werden. Beispiele für inhaltsbasierte Suchmaschinen im 3D Bereich sind in den Abbildungen 4.7 und 4.8 (entnommen aus dem ZfBB Artikel von Ina Blümel und Irina Sens [SB09], sowie aus einer ECDL Publikation aus dem Jahr 2009. [BBK*09]

Die Abbildung 4.9, entnommen aus der PROBADO ECDL Publikation aus dem Jahre 2010 zeigen ein Beispiel für eine 2D-Ergebnisvisualisierung aus dem 3D-Bereich, sowie eine Notenblattvisualisierung zur inhaltsbasierten Suche aus dem Musik Bereich. [BBC*10]

Im Rahmen der vorliegenden Arbeit wurde vor allem im Bereich des CORE-Layers (Datenmodell, Webservices), im Bereich des 3D-Repository (Datenmodell, Konvertierung von Altdaten, App für mobilen Zugriff auf PROBADO 3D) ein Beitrag geleistet. Details hierzu finden sich in Kapitel 6, 7 und 8, die eine genaue technische Beschreibung der Struktur und Funktionen von PROBADO, sowie der Umsetzung in den jeweiligen Teilbereichen enthalten. Das Projekt wurde Mitte 2011 abgeschlossen und an die Technische Informationsbibliothek Hannover übergeben, wo es im Rahmen des GetInfo Portals eingesetzt wird. Die Apps für den mobilen Zugriff auf PROBADO 3D befinden sich aktuell in der Testphase.

4.9 Vergleich der Systeme

Im folgenden Abschnitt wird versucht, nochmals einen kurzen Überblick über die wesentlichen Eigenschaften der betrachteten Systeme zu geben, sowie Gemeinsamkeiten und Unterschiede herauszuarbeiten. Da es sich bei REAP nicht um ein digitales Bibliothekssystem, sondern um ein im Rahmen eines solchen einsetzbaren Rechtemanagementsystem handelt, wird es hier nicht betrachtet.

Folgende Vergleichskriterien werden herangezogen:

- Unterstützte Dokumentenformate
- Systemarchitektur
- Projektspezifische Daten (Laufzeit, Verbreitung, Open-Source, Produktivsystem)

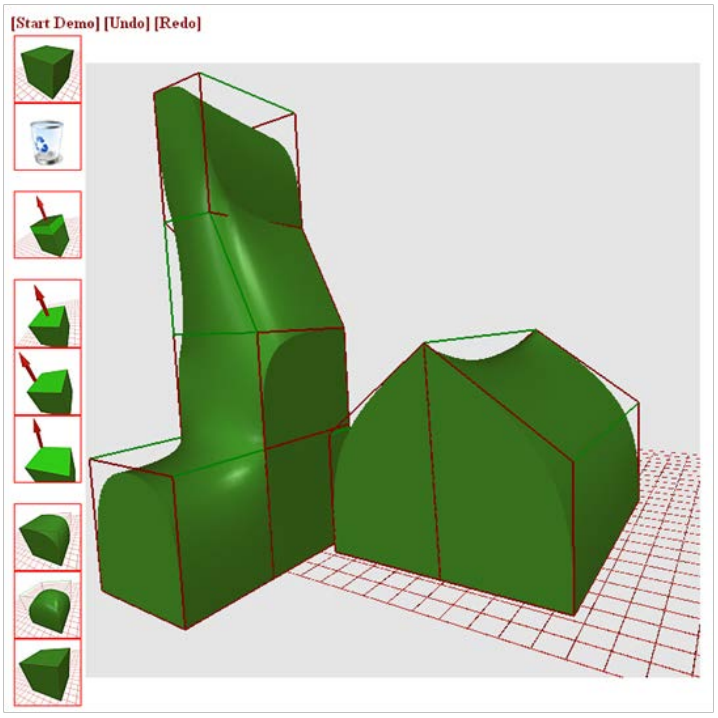


Abbildung 4.8: Anfrage aufgrund einer 3D-Skizze für inhaltsbasierte globale Suche. Quelle: [SB09], [BBK*09]

Formate	PROBADO	Delos DLMS	Bricks	Diligent	Marian	Fedora
Bücher (Bibliothekskatalog)	✓		✓	✓	✓	✓
Textuelle Dokumente allg.	✓	✓		✓	✓	✓
Video	✓	✓	✓	✓		✓
Audio	✓	✓	✓	✓		✓
3D Modelle	✓	✓	✓	✓		✓
Bilder	✓	✓	✓	✓		✓

Tabelle 4.1: Dokumentenformate

- Abfragemechanismen
- Benutzerinterfaces

4.9.1 Dokumentenformate

Die Tabelle 4.1 enthält eine Zusammenfassung darüber, welche Art von digitalen Objekten von welchem der beschriebenen digitalen Bibliothekssysteme unterstützt wird.

Anmerkungen zur Tabelle:

- Probado unterstützt sowohl Metadatensuche als auch inhaltsbasierte Suche (siehe 4.8), im Bereich der Metadatensuche wird jedes beliebige Format unterstützt.
- Da der Downloadlink für die Bricks Software aktuell nicht zur Verfügung steht, kann nicht geprüft werden, inwieweit die erwähnten Formate tatsächlich unterstützt werden.
- Diligent kombiniert Daten aus beliebigen Quellen und generiert daraus Berichte, die im Falle einer Änderung der zugrundeliegenden Daten automatisch aktualisiert werden.

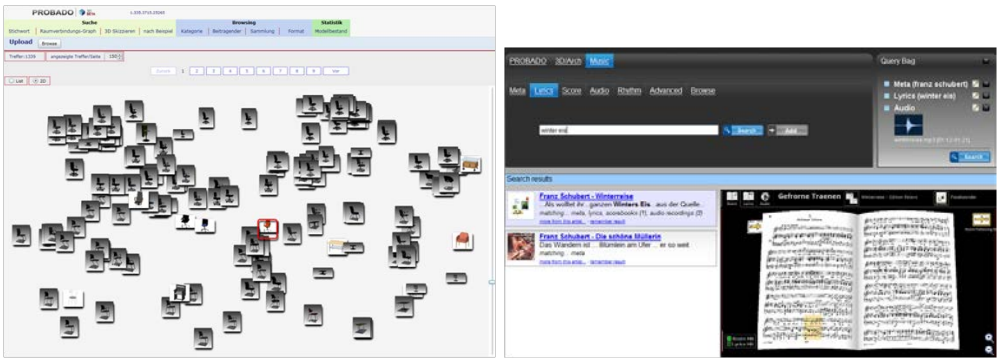


Abbildung 4.9: 2D Ergebnis Visualisierung. Quelle: [BBC*10]

Architektur	PROBADO	Delos DLMS	Bricks	Diligent	Marian	Fedora
Client\Server	✓	✓			✓	✓
Peer-To-Peer		✓	✓			
Grid				✓		
SOA	✓	✓	✓	✓	✓	✓

Tabelle 4.2: Systemarchitektur

- Die von Fedora eingesetzte „Content Model Architecture“ (CMA) erlaubt es beliebige Datenformate zu speichern. (Nicht zu verwechseln mit der inhaltlichen Erschließung).

Hier kann man eine Unterteilung in Systeme vornehmen, die den Fokus auf die Anbindung von Daten aus möglichst vielen verschiedenen Quellen legen (Fedora, Delos-DLMS, Diligent), sowie in solche, welche sich auf die optimale Erschließung bestimmter Daten spezialisiert haben (Marian, PROBADO).

4.9.2 Systemarchitektur

Dieser Abschnitt versucht einen Überblick über die in den verschiedenen digitalen Bibliothekssystemen eingesetzten Architekturprinzipien zu geben und Vergleiche zwischen den einzelnen Systemen zu ziehen.

Wie man aus der Tabelle 4.2 erkennen kann, überwiegt bei den betrachteten Systemen die klassische Client-Server Architektur. Alternative Architekturformen wie Peer-To-Peer Architektur und Grid Architektur werden nur im Rahmen der beiden Forschungsprojekte Bricks und Diligent eingesetzt.

4.9.3 Schnittstellen/Protokolle

Vor allem in Bezug auf die Themenbereiche Datenaustausch und Erweiterbarkeit spielen Schnittstellen nach außen und der Einsatz standardisierter Protokolle eine wichtige Rolle. Die Tabelle 4.3 bietet einen Überblick über die von den betrachteten Projekten unterstützten Protokolle. In Bezug auf die Schnittstellenprotokolle SOAP und WSDL ist noch anzumerken, dass die Implementierung von MARIAN bereits 1990 begann, zu einem Zeitpunkt, als diese noch kein Begriff waren.

4.9.4 Programmiersprache

Wie man aus Tabelle 4.4 ersehen kann, wurde ein Großteil der angebotenen Lösungen in Java entwickelt. Hauptgrund dafür dürfte der Wunsch nach Plattformunabhängigkeit sein. Die Motivation hinter der Implementierung von Teilen von Delos DLMS in C++ dürfte eine höhere Performance sein.

Anmerkungen zur Tabelle:

- Marian wurde zuerst in C++ entwickelt, eine Reimplementierung in Java wurde 1999 begonnen.

Protokoll	PROBADO	Delos DLMS	Bricks	Diligent (gCUBE)	Marian	Fedora
Proprietäre API				✓	✓	
SOAP	✓	✓	✓			✓
WSDL	✓	✓	✓			
OAI-PMH	geplant			✓		

Tabelle 4.3: Protokolle

Programmiersprache	PROBADO	Delos DLMS	Bricks	Diligent	Marian	Fedora
Java	✓	✓	✓	✓	✓	✓
C#	✓					
C++		✓			✓	

Tabelle 4.4: Programmiersprachen

- PROBADO: Für die Abfrage und die Ergebnispräsentation wurde ein auf HTML und Java-Script (J-Query) basierender, beliebig erweiterbarer Webrahmen geschaffen. Zur Visualisierung von 3D Suchergebnissen wird zusätzlich Microsoft Silverlight eingesetzt.

4.9.5 Unterstützte Plattformen

Die Tabelle 4.5 gibt einen Überblick darüber, unter welchen Betriebssystemen das jeweilige DLMS lauffähig ist. Mit Ausnahme von Delos DLMS und Marian, die Mac-OS nicht unterstützen, werden alle drei gängigen Betriebssysteme (Windows, Linux und Mac-OS) von sämtlichen verglichenen DLMS unterstützt. In bezug auf PROBADO muss noch erwähnt werden, dass die Plattformen Linux und Mac-OS vom CORE und dem 3D-Repository nicht unterstützt werden, da in diesen Bereichen MS-SQL Server als DBMS eingesetzt wird. Das Frontend, sowie das Musik Spezial Repository sind plattformunabhängig.

4.9.6 Projektbezogene Daten

Die Tabelle 4.7 versucht einen Überblick über projektbezogene Daten wie Projektlaufzeit, die Verbreitung der Lösung, ob es sich dabei um Open-Source handelt bzw. den Reifegrad des DLMS (Forschungsprototyp oder Produktivsystem) zu geben.

Anmerkungen zur Tabelle:

- Marian wurde im Rahmen des Bibliothekssystems von Virginia Tech eingesetzt, wird aber aktuell nicht mehr verwendet.
- PROBADO: Einsatz im Rahmen des GetInfo Portals der Technischen Informationsbibliothek Hannover ab Sommer 2011.

4.9.7 Zusammenfassung

Allgemein ist zu sagen dass, abgesehen von Fedora, sämtliche der betrachteten digitalen Bibliothekssysteme im Rahmen von universitären Forschungsprojekten entwickelt wurden. Allen Systemen gemein ist ein modularer, serviceorientierter Aufbau, vier von sechs Systemen setzen SOAP Schnittstellen ein. PROBADO wurde als einziges der betrachteten DLMS teils in C# entwickelt, alle anderen verwenden ausschließlich Java bzw. Java und C++. Der verbreitete Einsatz von Java ist sicher auch im Kontext der Plattformunabhängigkeit zu sehen, wobei abgesehen von PROBADO sämtliche DLMS zumindest Linux und Windows gleichermaßen uneingeschränkt als Plattform unterstützen. Ein weiterer wichtiger Punkt sind der Projektstatus und der Reifegrad der Software. Hier ist sicher Fedora in Bezug auf Verbreitung und Performance führend. Auch g-Cube (wurde im Rahmen des Diligent Projektes entwickelt) und Delos-DLMS haben eine aktive Community und eine aktuelle Website. MARIAN wurde in den 90-er Jahren im Rahmen des Bibliothekssystems von Virginia Tech eingesetzt und diente als Basis für die

Plattform	PROBADO	Delos DLMS	Bricks	Diligent	Marian (*)	Fedora
Windows	✓	✓	✓	✓	✓	✓
Linux	✓	✓	✓	✓	✓	✓
Mac-OS	✓		✓	✓		✓

Tabelle 4.5: Plattformen

Projektdaten	PROBADO	Delos DLMS	Bricks	Diligent (gCube)	Marian	Fedora
Projekt -laufzeit	2006-2011	2003-2007	2004-2007	2004-2007	1190-2000	Seit 2003
Status	F	F	F	P	P	P
Open Source		✓	✓	✓		✓
Verbreitung	Teststellung	Community	Community	Community	klein	groß

Tabelle 4.6: Projektdaten

Tabelle 4.7: Legende: F..Forschungsprojekt | P..Produktivsystem

Entwicklung des NDLTD Bibliothekssystems. Abgesehen von PROBADO und MARIAN sind sämtliche Lösungen als Open-Source anzusehen und mit Ausnahme von Bricks auch im Web für jedermann zugänglich. Im Bereich der unterstützten Datenformate ist auffallend, dass Fedora augrund seiner Content Model Architecture als einziges System die Ablage beliebiger Datenobjekte unterstützt, wobei hier auf den Unterschied zwischen Ablage und Erschließung hingewiesen werden muss.

Die grosse Stärke von PROBADO im Gegensatz zu vergleichbaren Systemen liegt in der konsequenten automatischen Erschliessung der Datenobjekte während des Einpflegevorgangs. Erst dadurch werden wichtige Metadaten gewonnen, die für die Kategorisierung des Objekts sowie die Suche unerlässlich sind. Insofern ist Content Model Architecture so zu verstehen, dass prinzipiell jegliche Art von Datenobjekten abgelegt werden kann, es jedoch dem Repositorybetreiber überlassen ist, die notwendigen Metadaten zur späteren Identifikation des Objektes zur Verfügung zu stellen - auf welchem Wege auch immer. Eine automatische Extraktion von Metadaten, die über Fileformat, Größe, Typ oder Einpflegedatum hinausgehen, wird von Fedora nicht unterstützt.

Kapitel 5

Eingesetzte Techniken

In diesem Kapitel wird ein kurzer Überblick über die im Rahmen des PROBADO Projektes eingesetzten Werkzeuge und Techniken gegeben. Ein wichtiger Punkt ist in diesem Zusammenhang der während des Projektes durchgeführte Technologiewechsel von Open-Source Lösungen zu Microsoft Produkten, dessen Ursachen im folgenden Abschnitt näher erläutert werden.

5.1 Technologiewechsel

Der erste Prototyp des CORE Systems wurde im Rahmen der Masterarbeit von Monika Alter erstellt. In dieser Arbeit mit dem Titel „Architektur eines offenen, verteilten Suchdienstes in nicht textuellen Dokument-Repositorien“ wurden der grundlegende Aufbau des CORE-Bereichs von PROBADO, sowie die technische Umsetzung spezifiziert. Weiters wurde unter Verwendung des *Apache Cocoon Frameworks* ein erster Prototyp erstellt. [Alt09] Im Zuge der Entwicklung des Prototyps traten vermehrt Probleme mit Cocoon auf. Diese waren unter anderem im Umfangreichtum des Frameworks begründet, das zu langen Einarbeitungszeiten der Projektmitarbeiter führte. Auch die immer wieder auftretenden Versionsinkompatibilitäten (jedes neue Cocoon-Release erforderte Anpassungen an bestehenden Codeteilen) erwiesen sich als problematisch. Der so entstandene Source-Code erreichte innerhalb kurzer Zeit einen unstrukturierten, schwer wartbaren Zustand. Aus diesem Grund wurde beschlossen, applikationsseitig auf eine stabilere, intuitiv zu bedienende Entwicklungsumgebung umzusteigen, deren Bekanntheitsgrad hoch genug ist, um Einarbeitungszeiten gering zu halten. Die Wahl fiel schließlich auf eine Kombination von Microsoft Produkten (*Visual Studio 2008, Microsoft Silverlight, Expression Blend*). Da vorgesehen war, sämtliche Datenbankzugriffe über *LINQ2SQL* abzuwickeln und zum Zeitpunkt dieser Entscheidung noch keine wirkliche Unterstützung von MySQL durch LINQ und Silverlight vorhanden war, wurde beschlossen, Microsoft SQL-Server als neues DBMS einzusetzen, mit der Option zu einem späteren Zeitpunkt auch andere DBMS anbinden zu können. Ausschlaggebend für die Entscheidung war unter anderem, dass Microsoft-Silverlight zu diesem Zeitpunkt im 3D-Bereich bereits erfolgreich im Einsatz war und man von den dort gemachten Erfahrungen profitieren wollte. Im Rahmen dieser Arbeit wurde der oben beschriebene Technologiewechsel praktisch vollzogen und bestehende Datenbankschemata (CORE,3D) angepasst, um den neuen Erfordernissen zu entsprechen. Zusätzlich wurden in Zusammenarbeit mit anderen Projektmitarbeitern die Webservices und Schnittstellenspezifikationen neu entwickelt bzw. adaptiert sowie neue Funktionalitäten (Recommendation Service, Zugriffsstatistiken) datenbankseitig vorbereitet. Weiters wurde der bestehende Webrahmen in Zusammenarbeit mit der TU Darmstadt um ein Interface und eine Ergebnisliste für die Anzeige von Suchresultaten aus der CORE Metadatensuche erweitert.

5.2 Eingesetzte Technologien

Im folgenden Abschnitt findet sich ein Überblick über die im Rahmen des Projektes eingesetzten Technologien. Diese werden, mit Ausnahme von SOA, SOAP und WSDL, auf die bereits in Kapitel 2 näher eingegangen wurde, im Anschluss beschrieben.

Liste der in Probado CORE eingesetzten Technologien:

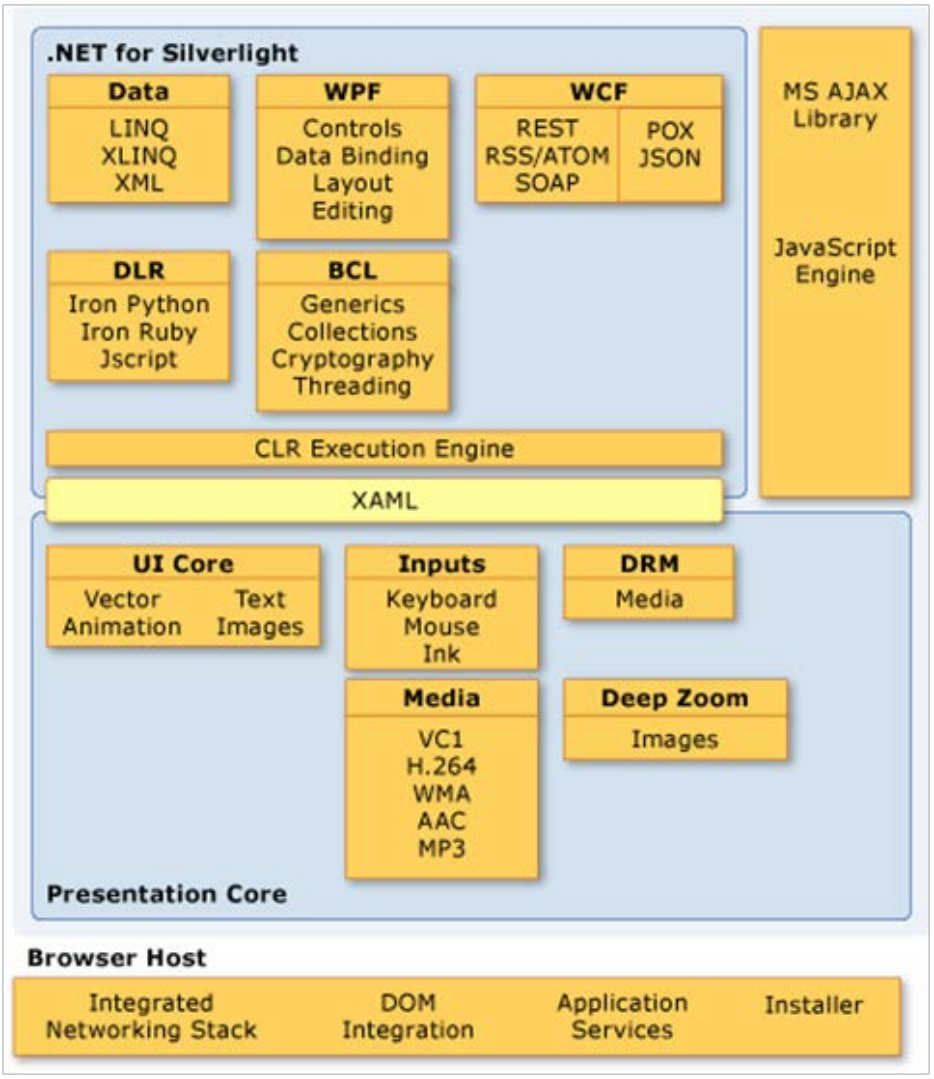


Abbildung 5.1: Architektur von Silverlight. Quelle: [Mic10]

- Microsoft Silverlight
- LINQ
- SOAP
- SOA
- WSDL

5.2.1 Microsoft Silverlight

Microsoft Silverlight ist eine Entwicklungsumgebung, die es ermöglicht „Rich Media Applications“ auf Webbrowsern und mobilen Geräten auszuführen. Es handelt sich dabei um ein kostenloses Browser-Plugin das für Windows und Mac-OS angeboten wird. [Mic10] Linux-Anwendern steht das von Novell entwickelte Moonlight zur Verfügung. Silverlight basiert auf dem .NET Framework und ist im Prinzip eine Plattform und Browser unabhängige Implementierung des *Windows Presentation Frameworks (WPF)*. [FK09] Aktuell liegt es in der Version 5.0 vor, die im Dezember 2011 von Microsoft freigegeben wurde. Die Abbildung 5.1 versucht einen groben Überblick über die Architekturprinzipien von Silverlight zu geben.

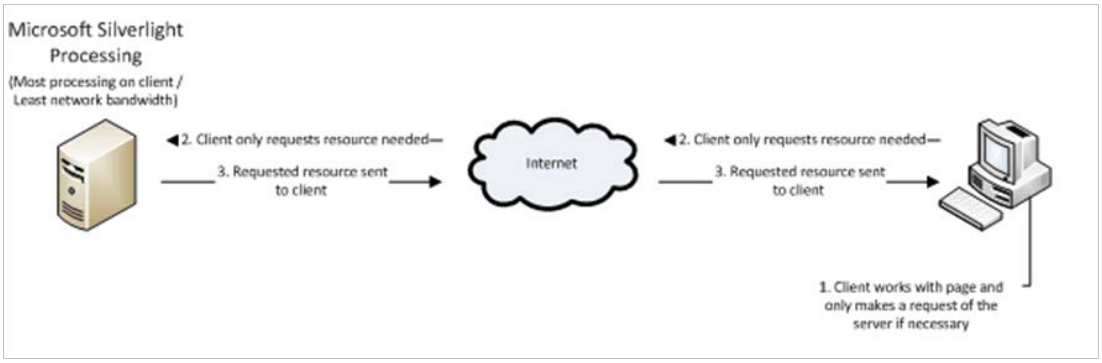


Abbildung 5.2: Silverlight Processing. Quelle: [Web10]

Silverlight Applikationen werden über ein Browser-Plugin im Webbrowser, auf dem Client ausgeführt. Das führt dazu, das dass die Rechenlast hauptsächlich auf Clientseite angesiedelt ist [Web10]. Dies wird durch Abbildung 5.2 bildlich dargestellt.

5.2.2 LINQ

LINQ kurz für Language Integrated Query ist ein Teil des .NET Frameworks der es ermöglicht Abfragen an beliebige Datenquellen direkt in .NET basierte Programmiersprachen einzubinden, wodurch eine Typüberprüfung bereits zur Compilezeit möglich wird. [Tec10c] Dabei wird eine standardisierte Abfrage-Syntax, angelehnt an SQL, („select“, „from“, „where“) verwendet, die auch komplexe Abfragen unterstützt. LINQ kann, in Abhängigkeit von der Datenquelle, in vier Hauptbereiche unterteilt werden [Fer08]:

- LINQ2Objects
- LINQ2ADO.NET
- LINQ2SQL
- LINQ2XML

Im Rahmen des PROBADO Projektes wurden LINQ2Entities (3D Bereich) und LINQ2SQL (CORE Bereich) eingesetzt.

LINQ2SQL *LINQ2SQL* ordnet das Datenmodell einer relationalen Datenbank einem Objektmodell zu. Dabei wird der Zustand der Objekte von LINQ überwacht, während die Applikation Änderungen daran vornimmt. Die Abfrage wird im Code der gewählten Programmiersprache geschrieben und bei der Ausführung von LINQ in SQL-Code übersetzt. Das Ergebnis der Datenbankabfrage wird von LINQ wiederum in Objekte übersetzt, die dann zurückgegeben werden. [MSD09] Abbildung 5.3, entnommen aus der MSDN Library, stellt den Ablauf einer LINQ2SQL Abfrage graphisch dar. Bei der Datenquelle muss entweder die generische IEnumerable (IEnumerable<T>)oder die abgeleitete IQueryable (IQueryable<T>) Schnittstelle unterstützen. Abgefragt werden die Werte z.B. durch eine foreach Schleife.

LINQ2Entities *LINQ2Entities* wurde von Microsoft 2008 als Teil des ADO .NET Entity Frameworks veröffentlicht, mit dem Ziel die Nachfolge von LINQ2SQL anzutreten. Dabei verfolgt das Entity Framework das Ziel, die Daten statt in einem Objektrelationsmodell in einem konzeptuellen Modell abzubilden. Daten können so anhand ihrer logischen Zusammengehörigkeit, tabellenübergreifend auf sogenannte „*Entities*“ gemappt werden. Die Abbildung 5.4 versucht einen groben Überblick über den Aufbau des Entity Frameworks zu geben.

Die Basis des Frameworks bilden die Datenquellen, deren Inhalte von den Datenprovidern zur Verfügung gestellt werden. In bezug auf Datenbanken wird hier aktuell nur MS-SQLServer unterstützt. Das darauf aufbauende Entity Daten Modell gliedert sich in die Ebenen der Konzeptuellen Schemadefinitionssprache (CSDL), der Speicher Schemadefinitionssprache (SSDL) und der Mapping Spezifikationssprache (MSDL). CSDL definiert Datenobjekte (Entities) und deren Beziehungen zueinander, SSDL

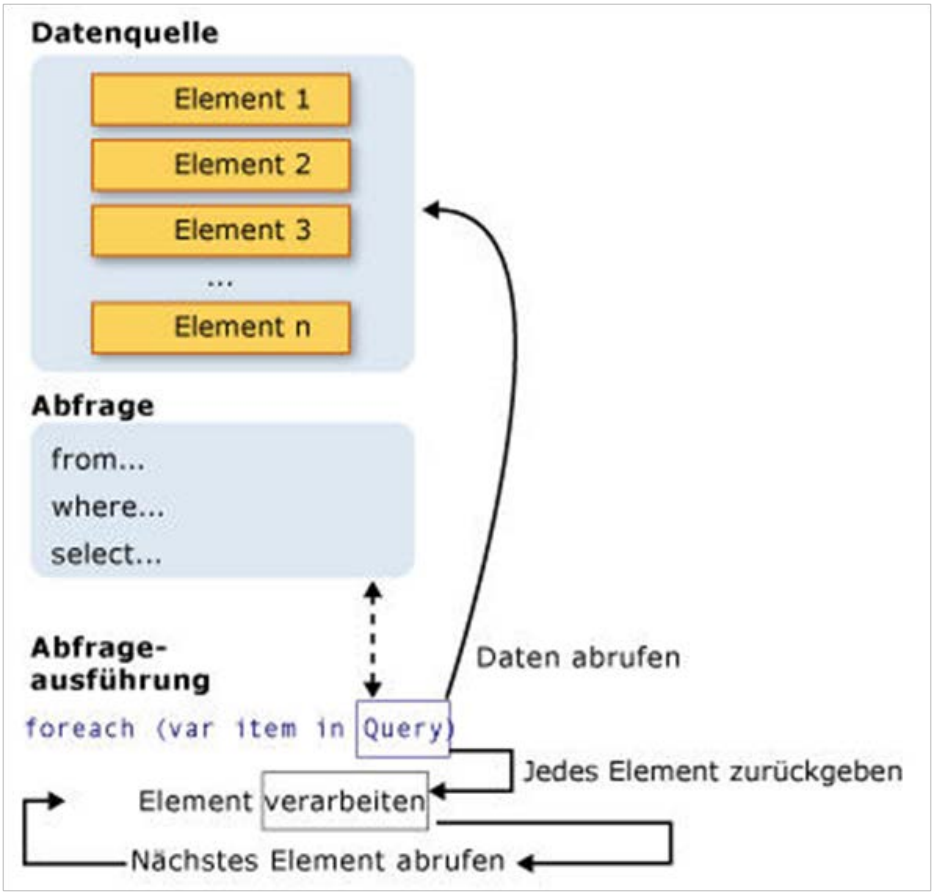


Abbildung 5.3: Ablauf einer LINQ2SQL Abfrage. Quelle: [MSD09]

definiert die Datenquelle (z.B. Datenbank), und MSDL legt das Mapping zwischen Objekten in der Datenquelle (z.B. Datenbanktabellen) und konzeptionellen Objekten fest. Der Entity Client verwaltet die Verbindung zur Datenquelle. Seine Funktionsweise ist vergleichbar mit einem Datenbankclient. Die darüber angesiedelte Schicht der Object Services führt Datenmanipulationen (insert, update, delete...) durch, die sowohl in LINQ2Entities als auch in Entity SQL (ein Derivat von Transact SQL) vorliegen können. Der Vorteil von LINQ2Entities gegenüber LINQ2SQL liegt in seiner größeren Flexibilität, und dem erweiterten Funktionsumfang. Dies wirkt sich positiv auf die Wartbarkeit der Applikation aus. Aufgrund der zusätzlichen Abstraktionsebene sollte LINQ2SQL im Bereich Performance in den meisten Fällen bessere Ergebnisse liefern. [Cru10]

5.3 Entwicklungsumgebungen

Die folgenden Entwicklungsumgebungen wurden im Rahmen der praktischen Arbeit eingesetzt:

- Microsoft Visual Studio 2008
- Microsoft Expression Blend 3
- SQL Server Management Studio

5.4 Datenbank

Als DBMS wurde zunächst die freie Express-Edition von MS-SQL-Server eingesetzt. Diese ist zwar für einen universitären Prototyp durchaus ausreichend, aufgrund der bestehenden Beschränkungen (2GB

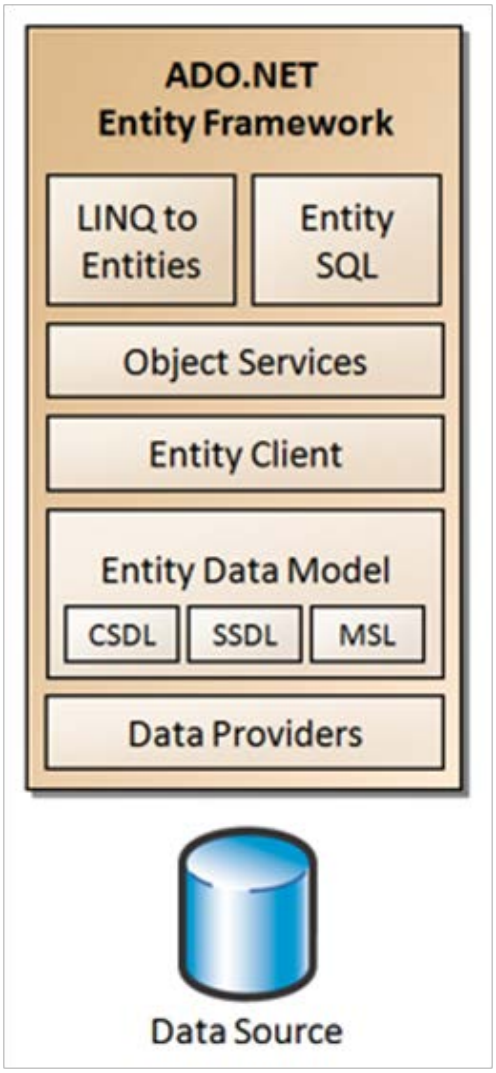


Abbildung 5.4: Aufbau des Entity Frameworks. Quelle: [Cru10]

RAM, Unterstützung für maximal einen Prozessor, eingeschränkte Funktionalität in Bezug auf automatische Sicherung, Import/Export von Daten usw.) wird aber empfohlen, für einen professionellen Einsatz auf eine kommerzielle Edition umzusteigen.

Kapitel 6

Umsetzung

In diesem und dem folgenden Kapitel werden die Struktur von PROBADO Core, die praktischen Ziele dieser Arbeit sowie deren technische Umsetzung detailliert beschrieben. Dabei gibt Kapitel 6 einen Überblick über Ausgangslage und Zielsetzung der Arbeit, beschreibt die einzelnen Systemebenen sowie die Kommunikationsvorgänge zwischen diesen Ebenen und geht schließlich detailliert auf den CORE-Bereich ein.

6.1 Ausgangslage und Zielsetzung

Zu Beginn des Diplomprojektes bestand folgende Ausgangslage: die Struktur des digitalen Bibliothekssystems war bereits in der vorangegangenen Projektphase definiert worden und stand fest. Da sich die Technologien, die zur Entwicklung des ersten Prototypen eingesetzt wurden, als wenig brauchbar erwiesen, war für den CORE-Bereich, sowie datenbankseitig auch für den 3D Bereich ein Technologiewechsel hin zu Microsoft-Produkten beschlossen worden. Ziel dieser Arbeit war es, in Zusammenarbeit mit dem PROBADO Projektteam, diesen Technologiewechsel durchzuführen und falls nötig Anpassungen und Erweiterungen an den bereits bestehenden Spezifikationen (Datenbankmodelle, WSDL-Protokolle) vorzunehmen. Der Prototyp musste neu erstellt werden und bereits bestehende Alt-Daten im 3D-Bereich mussten in die neue Datenbankstruktur übernommen werden. Weitere Zielsetzungen waren die Unterstützung mehrerer Sprachen durch das CORE und 3D Datenmodells sowie die Erweiterung der CORE Suche um Personalisierung von Suchabfragen und Benutzerfeedback. Das Webinterface für die CORE-Suche sollte erstellt und in den allgemeinen Webrahmen eingebunden werden. Um diese Ziele zu erreichen wurden folgenden Schritte durchgeführt:

- Genaue Analyse der vorhandenen Spezifikationen
- Überarbeitung und Anpassung der Spezifikation an die aktuellen Rahmenbedingungen (Schnittstellen, Datenbankmodell, Servicearchitektur, usw.)
- Implementierung eines Prototypen.

6.2 Aufbau von PROBADO

Eines der Hauptziele beim Design der Systemarchitektur von PROBADO war Flexibilität. Es sollten einerseits möglichst viele unterschiedliche Repositorien eingebunden werden können, andererseits sollte das Benutzerinterface für Suchabfragen und Resultatanzeige beliebig ausgetauscht werden können. Aus diesem Grund wurde ein lose gekoppeltes 3-Schicht Modell gewählt, das aus folgenden Ebenen besteht [Alt09]:

- Webinterface als Frontend Layer (Layer 1)
- Core Ebene als Middleware (Layer 2)
- Ebene der Spezial Repositorien (Layer3)

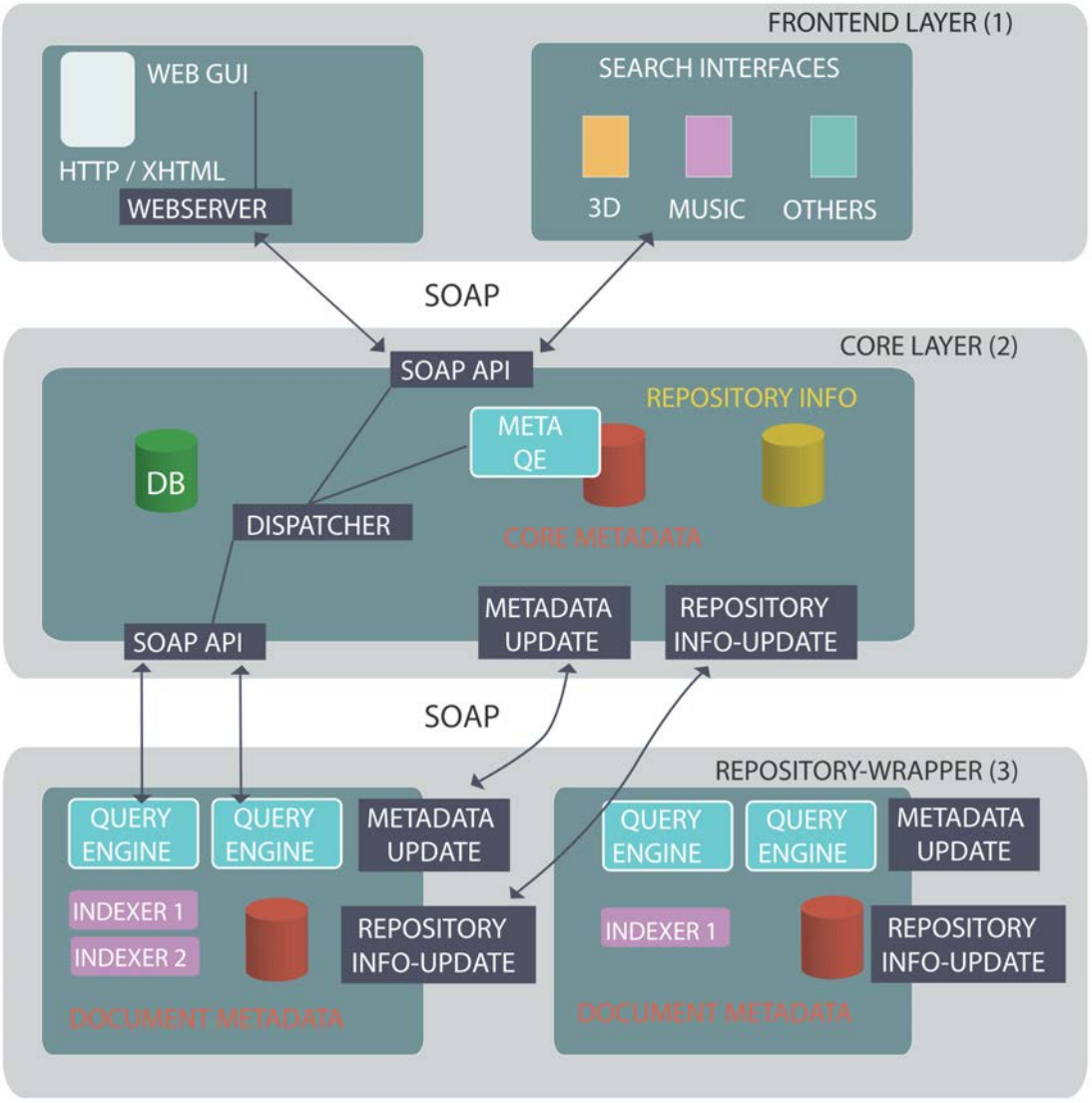


Abbildung 6.1: Grober Überblick über die PROBADO Systemstruktur. Quelle: [BBC*10]

Die Kommunikation zwischen den einzelnen Schichten erfolgt durch den Austausch von SOAP-Nachrichten über Webservices, die über ein WSDL Dokument eindeutig spezifiziert wurden. Abbildung 6.1, entnommen aus der ungekürzten ECDL-Publikation aus dem Jahr 2010 [BBC*10], zeigt den Aufbau von PROBADO sowie die Interaktion zwischen den einzelnen Ebenen.

Suchabfragen werden grundsätzlich über das jeweilige Webinterface abgesetzt und direkt an den CORE gestellt. Je nach Art der Anfrage wird diese entweder im CORE ausgeführt oder an ein/mehrere Spezialrepositorien weitergeleitet. Das Spezialrepository führt die Suchanfrage aus und übermittelt das Ergebnis an den CORE, der sie wiederum an die anfragende Stelle weiterleitet. Der aktuelle Prototyp hat jeweils ein 3D und ein Musik Repository angebunden. Die folgenden Abschnitte enthalten eine detaillierte Beschreibung der PROBADO Layer.

6.2.1 Frontend Layer

Das Frontend Layer bildet die Schnittstelle zwischen Benutzer und System. Hier werden Suchanfragen an den CORE gestellt und Suchresultate präsentiert. Die Kommunikation mit dem CORE erfolgt mittels SOAP Nachrichten über ein Webservice. Für das Webinterface des PROBADO Prototyps wurde von der Universität Bonn ein Webrahmen entwickelt, in den die Interfaces des CORE sowie der Spezialbereiche eingebettet wurden. Dieser Webrahmen wurde in der Endphase des Projektes an den CORE Bereich

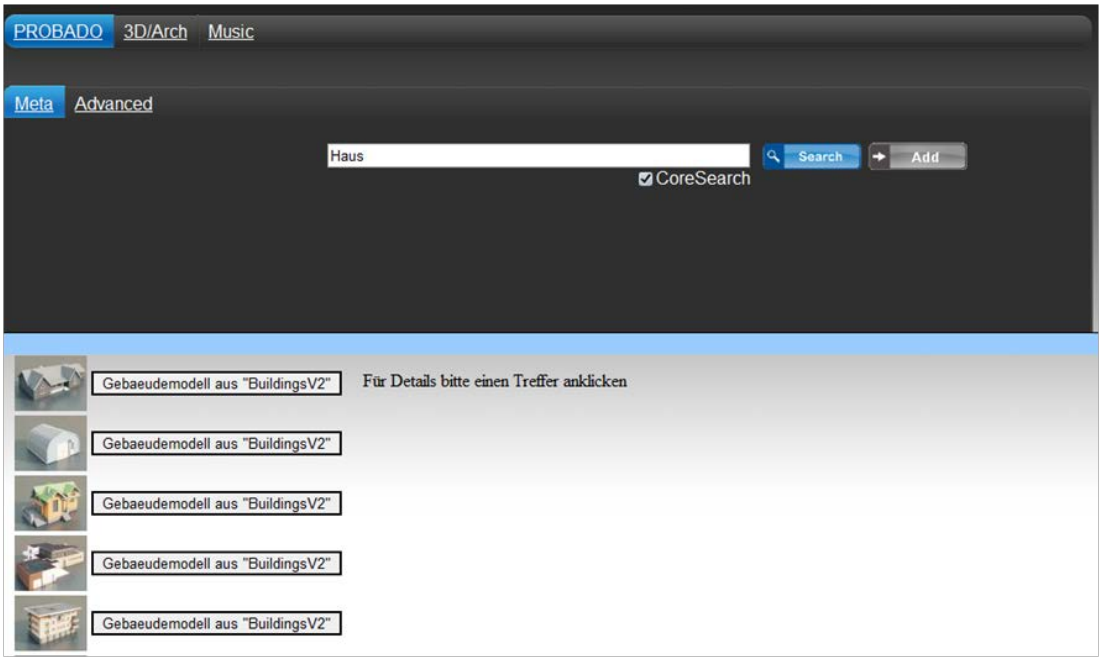


Abbildung 6.2: Webinterface einfache CORE Suche mit Ergebnisliste

übergeben und im Rahmen dieser Arbeit in Zusammenarbeit mit der TU Darmstadt um ein Suchinterface für den CORE-Bereich erweitert.

Suchanfragen werden über das Webinterface an den CORE bzw. ein Spezial-Repository gestellt. Unabhängig davon, ob die Suchanfrage von einem Spezialclient gestellt wurde, setzen sich die in der Ergebnisliste präsentierten Suchergebnisse stets aus CORE-Metadaten zusammen. Erst bei Anklicken eines Links aus der Ergebnisliste wird eine detaillierte Spezialanzeige geöffnet. Zur Erstellung des Webrahmens wurden HTML und Java Script eingesetzt. Während sich die Musik-Spezialsuche HTML und Java Script (jQuery [1]) einsetzt basiert die 3D Spezialsuche zum Beispiel auf Silverlight[2] und .NET. Die Oberfläche des CORE Suchinterfaces ist wie der Webrahmen mit HTML und Java-Script gestaltet. Im folgenden Abschnitt wird das Suchinterface des CORE Bereichs genauer beschrieben.

Suchinterface CORE Bereich Für den CORE Bereich wird eine Volltextsuche in den CORE-Metadaten (Aktenreiter Meta) sowie eine Spezialsuche innerhalb spezieller Metadatenfelder angeboten. Eine detaillierte Beschreibung des Suchvorganges findet sich in Abschnitt 6.9.

Die CORE Volltextsuche (siehe Abbildung 6.2) besteht aus einem einfachen Textfeld zur Eingabe des Suchbegriffs (der Suchbegriffe), einem Suchknopf sowie einem Knopf zum Hinzufügen des Suchbegriffs zum sogenannten “Querybag” für bereichsübergreifende Suchabfragen. Ist die Checkbox “Core” angehakt, dann wird die Suchabfrage ausschließlich auf der CORE-Datenbank ausgeführt und nicht an angeschlossene Spezialrepositorien weitergeleitet.

Die Suchergebnisse werden unterhalb des Suchbereichs in tabellarischer Form dargestellt, wobei jeder Eintrag aus einem Vorschaubild (wenn vorhanden) und einem Titel besteht (siehe Abbildung 6.3). Klickt man auf einen Eintrag in der Ergebnisliste wird die zugehörige Detaildarstellung geöffnet, die neben detaillierten Informationen zum Suchergebnis auch einen Link zum Provider (Downloadlink) enthält. Abbildung 6.4 zeigt die Detaildarstellung eines 3D-Modells aus der Suchergebnisliste.

Die erweiterte CORE Metadatensuche (dargestellt in Abbildung **??**) ermöglicht eine gezielte Metadatensuche in ausgewählten Feldern sowie eine benutzerdefinierte Reihung der Ergebnisse. Die Ergebnisliste verhält sich analog zur einfachen CORE-Suche (CORE Volltextsuche).

[1] http://jquery.com/
[2] http://www.silverlight.net/

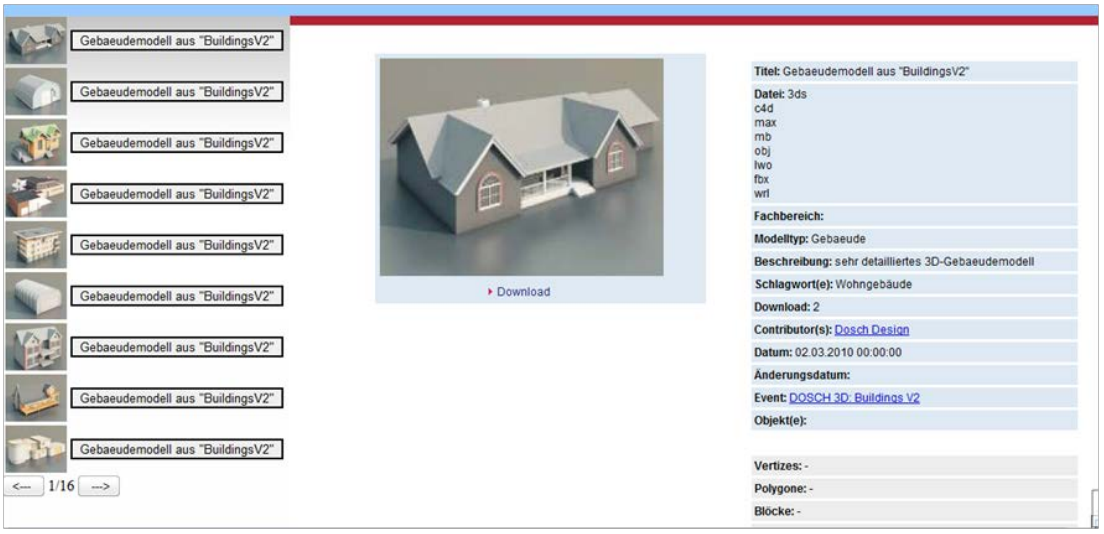


Abbildung 6.3: Detaildarstellung eines Ergebniseintrags

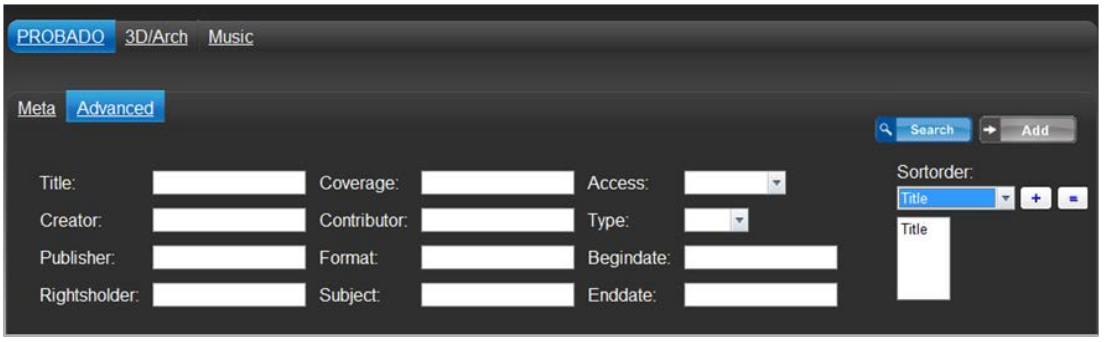


Abbildung 6.4: Webinterface erweiterte CORE Metadatensuche

6.2.2 Core Layer

Die CORE Schicht (Layer 2) erfüllt zum einen administrative Aufgaben, indem sie Benutzer und Repositorien verwaltet, zum anderen dient sie als Dispatcher für Suchanfragen. Sämtliche Anfragen werden zunächst an den Core gestellt, von wo aus sie dann entweder lokal ausgeführt werden (Suche innerhalb der Core Metadaten) oder an die entsprechenden Repositorien weitergeleitet werden. Die Suchergebnisse der Spezialrepositorien werden ebenfalls über den Core an den Benutzer zurückgeliefert. Um die Suche zu beschleunigen wird ein Teil der Metadaten aus den Spezial-Repositorien in den Core gespiegelt. Im Falle einer bereichsübergreifenden Anfrage werden die Suchergebnisse aus den Spezialrepositorien im CORE, entsprechend ihrer Relevanz gereiht, zu einer Ergebnisliste zusammengefügt, die dann an die anfragende Stelle rückübermittelt wird.

6.2.3 Repository Layer

Der Repository Layer bildet die Ebene der Spezial Repositorien. Diese sind über Webservices und SOAP Nachrichten lose an den CORE angebunden. Um mit dem CORE kommunizieren zu können, muss sich ein externes Repository zunächst einmalig registrieren und erhält einen eindeutigen Repository-Identifier zugeteilt. Dieser Identifier identifiziert das Repository in sämtlichen weiteren Kommunikationsvorgängen. Die Kommunikation kann dabei in zwei separate Vorgänge eingeteilt werden. Zum einen kann optional ein Sub Set an Metadaten an den CORE übertragen werden. Dieses Vorgehen ermöglicht eine beschleunigte Metadatensuche im CORE Bereich. Eine Verzweigung auf das Spezialrepository ist somit nur dann notwendig, wenn ein spezielles Suchverfahren verwendet werden soll oder bereichs-

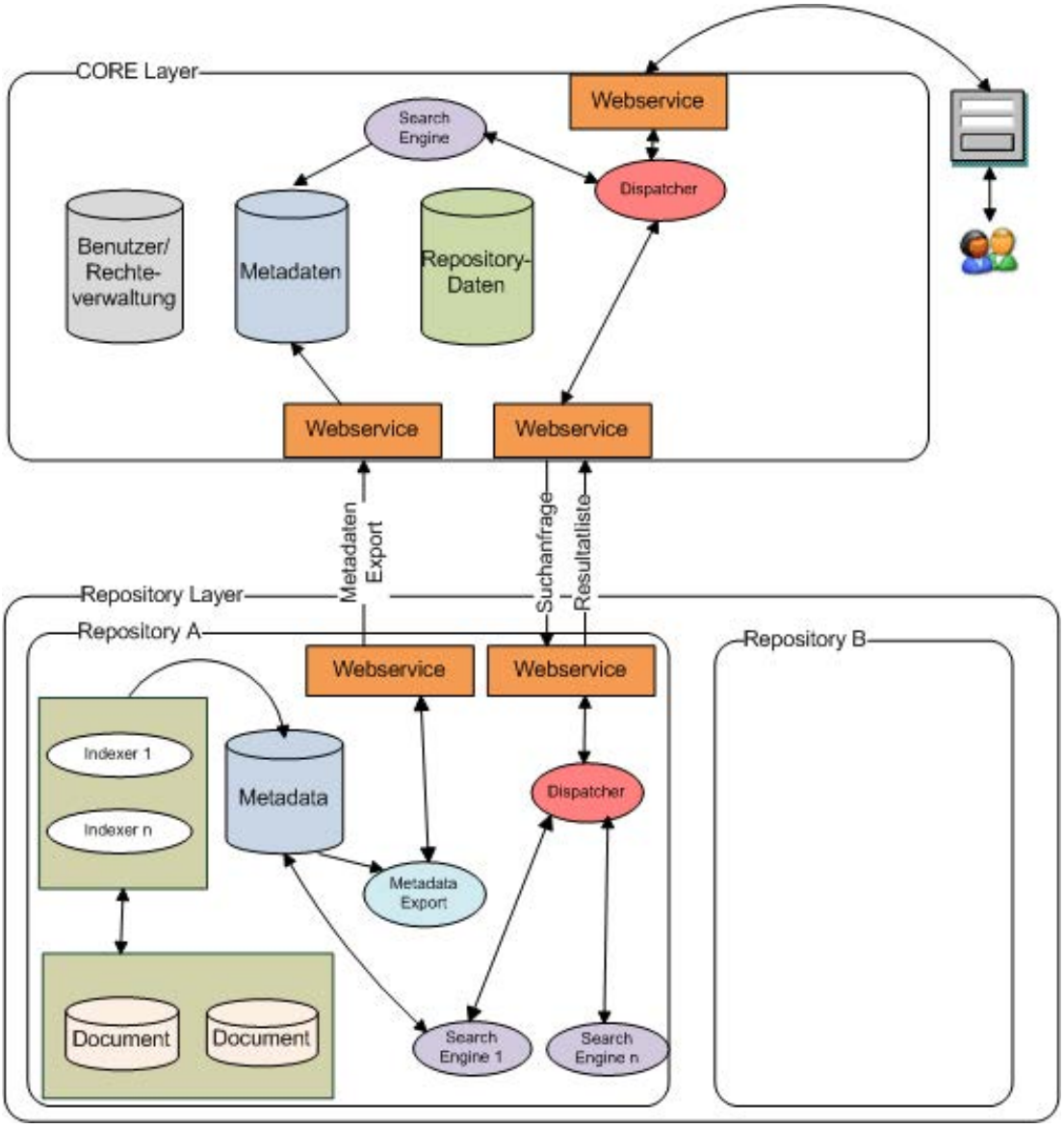


Abbildung 6.5: Kommunikationsvorgänge zwischen CORE und Spezialrepositorien

spezifische Metadaten abgefragt werden sollen. Zum anderen werden bereichsübergreifende oder an bestimmte Repositorien adressierte Suchanfragen an diese weitergeleitet.

Abbildung 6.5 bietet einen groben Überblick über diese Kommunikationsvorgänge.

DC-Element	Beschreibung
DC_identifier	Eindeutiger Identifier, enthält die URN oder DOI eines Dokuments. Wird in PROBADO einerseits zur eindeutigen Identifikation eines Repositories, andererseits zur eindeutigen Identifikation eines Dokuments verwendet. Muss verpflichtend vorhanden sein.
DC_description	Enthält eine textuelle Kurzbeschreibung eines Objektes. Das Element ist optional kann aber nur einmal vorkommen. Jedes Dokument hat also nur eine optionale Beschreibung.
DC_language	Bezeichnet die Sprache des Dokumentes. Laut Spezifikation sollen hier ISO 639-2 Sprachcodes zum Einsatz kommen. (http://www.loc.gov/standards/iso639-2/php/code_list.php). Die Bezeichnung in der lokalen Sprache sowie die englische Bezeichnung sind optionale Attribute.
DC_date	Kann beliebige Datumsangaben beinhalten, die jeweils optional ein Kategorie Attribut enthalten können. Das Element ist optional und kann beliebig oft vorkommen.
DC_creator	Dieses Attribut kann Urheber oder ganz allgemein Kontributoren bezeichnen. Das Attribut contributor\role legt fest, um welche Art des Beitrags es sich handelt.
DC_subject	Wird für die Kennzeichnung von Schlagwörtern für die Klassifizierung der Dokumente verwendet.
DC_format	Beinhaltet den MIME-Type des Dokuments.
DC_type	Legt das Genre (z.B. Musik oder 3D) des Dokuments fest.
DC_pulisher	Optionales Element – bezeichnet die Institution/Person die das Repository/Dokument veröffentlicht.
DC_title	Bezeichnet den Titel eines Dokuments (Z.B. Werktitel oder Objekttitel). Ist verpflichtend und kann pro Datensatz maximal einmal vorkommen.
DC_coverage	Räumliche Informationen zum Inhalt eines Dokuments. (Z.B. geographische Lage)
DC_rights	Informationen über Berechtigungen in Zusammenhang mit einem Objekt. Über das Attribut *accessrights* können Zugriffsrechte festgelegt werden. Das Attribut *license* bildet Lizenzinformationen ab.

Tabelle 6.1: Metadaten

6.3 Metadaten im CORE

Bei den im CORE abgelegten Metadaten muss zunächst zwischen repositoryspezifischen und dokumentspezifischen Metadaten unterschieden werden. Um Anfragen gezielt weiterleiten zu können müssen dem CORE Informationen (Metadaten) über die angeschlossenen Repositorien zur Verfügung stehen. Um eine beschleunigte Metadatensuche anbieten zu können, wird ein Teil der in den Spezialrepositorien abgelegten Metadaten in den CORE gespiegelt. Dies ist vor allem bei unspezifischen, eher allgemein gehaltenen Suchanfragen mit einfachen Suchbegriffen ein Vorteil. So hat der Benutzer die Möglichkeit, zunächst bereichsübergreifend die CORE Datenbank zu durchsuchen. Er erhält schnell ein Ergebnis und kann seine Anfrage später anhand der erhaltenen Ergebnisse verfeinern bzw. bei Bedarf auf ein bereichsspezifisches Spezialinterface wechseln. Da sich die in den einzelnen Spezialrepositorien abgelegten Metadaten doch deutlich unterscheiden musste hier zunächst eine Schnittmenge gemeinsamer Daten definiert werden. Man entschied sich bei der Auswahl der CORE Metadaten einen Teil des Dublin Core Schemas zu verwenden. [Alt09] Diese Schnittmenge wurde im Zuge der Reimplementierung noch um einige Elemente erweitert. Detaillierte Informationen finden sich in der Liste der aktuell abgeglichenen Metadaten. 6.1 Abgesehen von den neu hinzugekommenen Elementen DC_coverage und DC_rights ist sie ident mit der im Rahmen der Diplomarbeit von Monika Alter erarbeiten Liste. [Alt09]

6.4 Datenbankmodell

Das aktuelle CORE Datenbankmodell gliedert sich in vier Teilmodelle, wobei die Teilmodelle 1,3 und 4 im Rahmen dieser Arbeit entwickelt wurden. Das Metadatenmodell ist eine Adaption des von Monika Alter entwickelten Basis Modells. [Alt09]

1. Allgemeine Stammdaten: enthält ein Datenmodell zur Speicherung von Stammdaten in Listen oder Katalogform.
2. Metadaten: enthält Metadaten über die angeschlossenen Repositorien und deren Inhalt. Hier werden sowohl beschreibende Metadaten zu den gespeicherten Informationsobjekten (Bezeichnung, Beschreibungen, Schlagworte usw.) als auch strukturelle Metadaten (am Beispiel eines 3D Modells, die Zusammenhänge zwischen Gesamt und Teilmodellen), sowie administrative Metadaten (Datentypen der abgelegten Modelle, Erstellungszeitpunkt, Dateigröße usw.) abgelegt.
3. Administration: Tabellenstruktur für eine Benutzer und Rechteverwaltung.
4. Ablaufdaten: Modell zur Protokollierung von Suchanfragen.

Dabei bildet Modell 1 (allgemeine Stammdaten) die Basis. Es muss zuerst angelegt werden, da alle anderen Teilmodelle darauf referenzieren. Zwischen den Modellen 2-4 bestehen jedoch keine Abhängigkeiten. Hier wurde bewusst auf den Einsatz von Fremdschlüssel-Beziehungen verzichtet, um diese Bereiche bei Bedarf austauschen zu können. In den folgenden Unterabschnitten findet sich eine detailliertere Beschreibung der einzelnen Teilmodelle.

6.4.1 Allgemeine Stammdaten

Das Modell für allgemeine Stammdaten (Enumerationsdaten, Listendaten) wurde aus zwei Gründen geschaffen:

1. Mehrsprachigkeit
2. Schaffung einer allgemeinen Struktur zur Ablage vorgegebener Stammdaten, die wenige Attribute aufweisen, über eine begrenzte Anzahl von Einträgen verfügen, und sich über die Zeit kaum ändern. (Beispiel: Texte in Auswahllisten)

Diese Daten sollen gesammelt in einer einzigen Struktur abgelegt werden können, sodass nicht für jede inhaltlich zusammengehörige Liste von Daten (z.B. Repositorytypen, Länder usw.) eine eigene Tabelle angelegt werden muss, die Daten aber trotzdem nicht hardcoded vorliegen und dynamisch erweitert bzw. geändert werden können. Das Datenbankmodell setzt sich aus drei Tabellen und einem View zum leichteren Zugriff auf sprachabhängige Texte zusammen. Das Modell weist eine Baumstruktur auf. Eine einfache Liste besteht aus einem Basisknoten (Definition der Liste selbst) und mehreren Kindern (den eigentlichen Listeneinträgen). Dies wird durch Abbildung 6.6 dargestellt.

Identifikation

Jede Liste hat mit der Spalte LISTID einen eindeutigen numerischen Identifier, der den Eintrag auf der Datenbank identifiziert. Dieser Identifier wird beim Einfügen des Datensatzes in die Datenbank automatisch generiert und ist auf jeder Datenbank unterschiedlich. Zur Identifikation über Datenbankgrenzen hinweg dient der lokale-Identifier (LOCALID), der in Zusammenhang mit der den Spalten PARENTID, LANGUAGE und dem Gültigkeitsbereich (VALIDFROM, VALIDTO) eindeutig ist. Eine erste Version dieses Modells wurde bereits in der dieser Arbeit vorangegangenen Seminar/Projektarbeit [Foi09] geschaffen und im Zuge dieser Arbeit adaptiert.

Länderabhängigkeit

Eine Liste kann sowohl global gültig, als auch länderabhängig sein. Beide Varianten können nebeneinander existieren. Die globale Liste muss verpflichtend vorhanden sein.

- Global gültig: Feld LANGUAGE in LISTDEF ist NULL

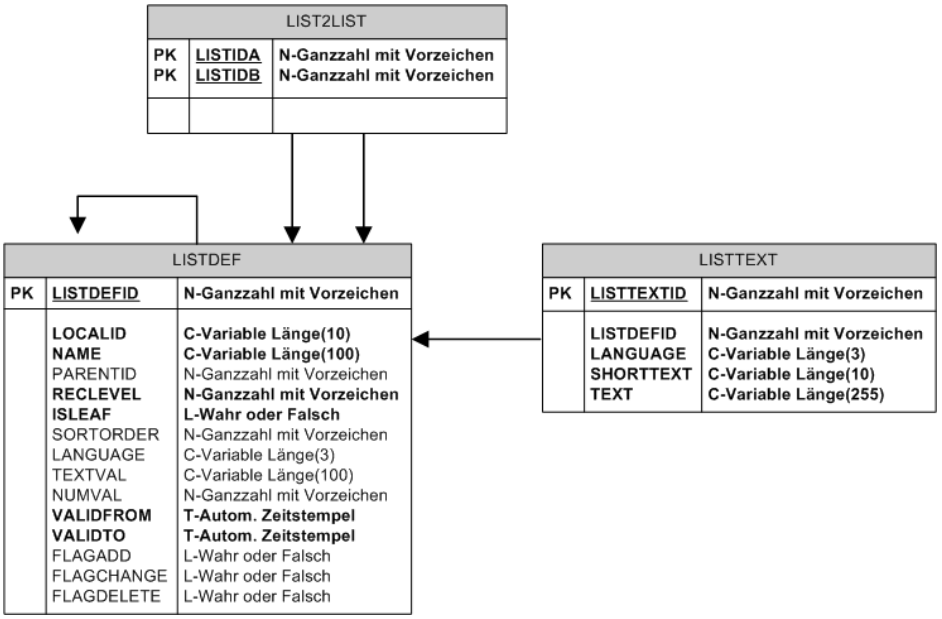


Abbildung 6.6: Listenmodell

- Länderabhängig: Feld LANGUAGE in LISTDEF enthält eine Länderkennung (z.B. das Kürzel de für Deutschland).

Es können parallel sowohl eine globale Liste als auch für einige Länder spezifische Listen angelegt werden. Gibt es für das dem aktuell angemeldeten Benutzer zugeordnete Land (Sprache) keine Liste, wird die globale verwendet. Die Länderabhängigkeit von Listen macht dann Sinn, wenn Art und Inhalt der Einträge je nach Land variieren und hat nichts mit der Übersetzung von Texten zu tun. Eine Liste, die Postleitzahlen enthält, wäre z.B. länderabhängig.

6.4.2 Repositorien Datenmodell

Darunter ist das Datenbankmodell für Repository Metadaten im CORE zu verstehen. Dieses beinhaltet Informationen zu den Repositorien (Tabelle REPOSITORY), den angeschlossenen Suchmaschinen (Tabelle QUERYENGINE) und den von diesen unterstützten Datentypen (Tabelle DATATYPE) sowie den von den Repositorien abgedeckten Bereichen (Tabelle REPOSITORY2SUPPORTEDTYPES) wie z.B. 3D oder Musik. Die unterstützten Sprachen werden in der Tabelle REPOSITORY2LANGUAGES abgelegt. Da die hier abgelegten Daten von den angeschlossenen Repositorien über ein Webservice bekanntgegeben werden und eine möglichst große Fehlertoleranz sichergestellt werden soll, ist das Modell so gestaltet, dass es möglichst wenige verpflichtende Felder enthält.

Das Modell ist im Detail wie folgt aufgebaut:

1. Zu jedem Repository stehen folgende Attribute zur Verfügung::

   - REPOSITORY_IDENTIFIER: (Verpflichtend) Eindeutige Kennung des Repositories. Um an PROBADO angeschlossen zu werden, muss ein Repository zunächst einen Repository Identifier beantragen.
   - TITEL: (Verpflichtend) Eindeutige Bezeichnung des Repositories.
   - REP_DESCRIPTION: Beschreibung des Repositories.
   - SEARCH_SOAP_ADDRESS: Addresse des SOAP-Interfaces (für Volltext oder Metadatensuche im Repository).
   - REP_OWNER: Betreiber des Repositories.
   - AI_AVAILABLE: Status des Repositories. Mögliche Werte: ONLINE, TIMEOUT, OFFLINE

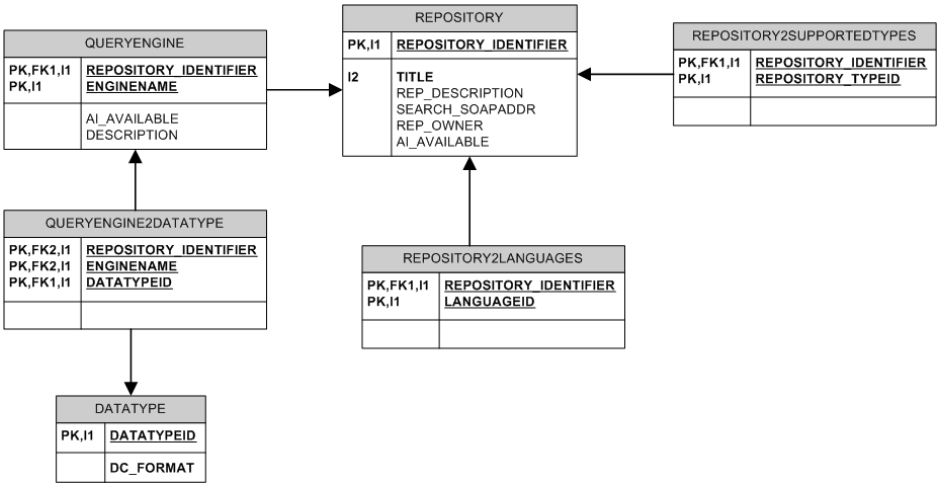


Abbildung 6.7: Repository Datenbankmodell

2. Jedes Repository kann optional die von ihm unterstützten Sprachen und Datentypen bekanntgeben (Tabellen REPOSITORY2LANGUAGES und REPOSITORY2SUPPORTEDTYPES).

3. Jedes Repository kann optional eine oder mehrere Suchmaschinen bekanntgeben (Tabelle QUERYENGINE), die wiederum mit den von ihnen unterstützten Datentypen verknüpft sein können (Tabellen QUERYENGINE2DATATYPE, DATATYPE)..

Abbildung 6.7 gibt einen Überblick über die aktuelle Tabellenstruktur.

Das Modell basiert auf einem ersten, bereits von Monika Alter entwickelten Datenmodell [Alt09], das im Rahmen dieser Arbeit folgendermaßen verändert wurde: Die Tabellenstruktur wurde grundsätzlich vereinfacht und so umgestaltet, dass

1. nicht verwendete Spalten entfielen.

2. die Tabellen an die Bedürfnisse des in Zusammenarbeit mit der TU-Darmstadt programmierten Webservices angepasst wurden.

3. die Bezeichnungen der Tabellen, Spalten und Schlüssel an ein einheitliches Schema angepasst wurden.

4. in Bezug auf die registrierten externen Repositorien nur ein minimaler Kerndatensatz als verpflichtend anzusehen ist.

5. in den WSDL-Spezifikationen möglichst wenige Einschränkungen vorgenommen werden müssen. Von Suchmaschinen unterstützte Datentypen werden z.B. als Text und nicht als Auswahlliste definiert und abgelegt, um administrativen Aufwand beim Hinzufügen eines neuen Datentyps zu vermeiden.

6. das administrative Modell (welche Repositorien mit welchen Suchmaschinen stehen zur Verfügung) vom beschreibenden/strukturellen Modell (welche Metadaten zu Objekten sind in welchem Repository vorhanden und wie hängen sie zusammen) abgetrennt werden kann. Aus der Tabelle REPOSITORY wurden nicht verwendete Spalten entfernt. Die Spalte DC_Type wurde in eine eigene Tabelle REPOSITORY2SUPPORTEDTYPES ausgelagert, da ein Repository laut Definition mehr als eine Art von Inhalten enthalten kann (z.B. Metadaten und 3D Objekte). Die Tabelle QUERYFORMAT entfiel und die unterstützten Datenformate wurde statt mit dem Repository mit der Queryengine verbunden.

6.4.3 Metadatenmodell

Das Metadatenmodell bezeichnet das aktuelle Modell der im CORE abgelegten Metadaten. Dabei handelt es sich um bereichsübergreifende Metadaten, die für beide Projektbereiche (3D und Musik) von

Bedeutung sind, sowie um Stammdaten zur Identifikation und Beschreibung der angeschlossenen Repositorien. Bei der Gestaltung des Modells wurde versucht möglichst nah am ursprünglichen von Monika Alter entwickelten CORE-Datenbankmodell zu bleiben. [Alt09]

Das aktuelle Modell ist wie folgt aufgebaut:

1. Die Tabelle DOCUMENT_METADATA enthält zu jedem, dem CORE vom jeweiligen Quellrepository bekanntgegebenen Objekt, genau einen Datensatz mit Metadaten zu diesem Objekt. Folgende Attribute können abgelegt werden:

   - DOCMETADATAID (verpflichtend): Eindeutiger Index für Volltextsuche - wird beim Einfügen des Datensatzes automatisch generiert.
   - REPOSITORY_IDENTIFIER (verpflichtend): Eindeutige Kennung des Quellrepository.
   - DOCUMENT_IDENTIFIER (verpflichtend): Eindeutige Kennung des gespeicherten Objektes. Bildet gemeinsam mit dem REPOSITORY_IDENTIFIER den Primärschlüssel der Tabelle.
   - PARENT_DOCIDENTIFIER (verpflichtend): Eindeutige Kennung des Datensatzes im Ursprungsrepository.
   - LINK_TO_PRESENTATION (verpflichtend): Link auf das Objekt.
   - DOCUMENT_TITEL (verpflichtend): Titel des Objektes.
   - PERSISTENT_IDENTIFIER (optional): Z.B. Unified Resource Identifier (URI) oder Digital Object Identifier (DOI) des Objektes.
   - DESCRIPTION (optional): Textuelle Beschreibung des Objektes.
   - DOCUMENT_TYPE (optional): Typ des Objekts. Aktuell mögliche Typen sind 3D und MUSIC.
   - CREATOR (optional): Urheber/Autor des Objektes.
   - PUBLISHER (optional): Publisher des Objektes.
   - RIGHTSHOLDER (optional): Rechteinhaber des Objektes.
   - LANGUAGEID (optional): Link auf die Liste der zur Verfügung stehenden Sprachen. Sprache, in der die Metadaten des Objektes vorliegen.
   - COVERAGE (optional): Örtlicher oder zeitlicher Bezug zum Objekt.
   - ACCESSRIGHTS (optional): Referenz auf Zugriffsrechte - Platzhalter für zukünftige Erweiterungen.
   - LINK_TO_THUMBNAIL (optional): Im Falle von 3D-Objekten/Grafiken ein Link auf ein Vorschaubild.

2. Weitere mit dem Objekt verknüpfte optionale Metadaten können in Form einer 1:n Beziehung vorliegen und werden daher, wenn vorhanden, in eigenen Tabellen abgelegt. Das Modell beinhaltet hierzu folgende Tabellen:

   - DOCUMENT2DATE: Ablage für das Objekt wichtiger Zeiten bzw. Zeiträume (z.B. Entstehungszeitraum).
   - DOCUMENT2DATATYPE: Format(e) in denen das Objekt vorliegt (z.B. mp3, cad, usw..)
   - DOCUMENT2CONTRIBUTOR: Kontributoren des Objektes.
   - DOCUMENT2SUBJECT: Schlagworte und Begriffe die mit dem Objekt in Verbindung stehen.

Abbildung 6.8 zeigt das aktuelle CORE Metadatenmodell.

Folgende Änderungen zum Ursprungsmodell von Monika Alter [Alt09] wurden vorgenommen:

- Auslagerung von Enumerationsdaten ins Listenmodell
- Umbennung der Datenbanktabellen und Spalten anhand des neuen Bennungsschemas.

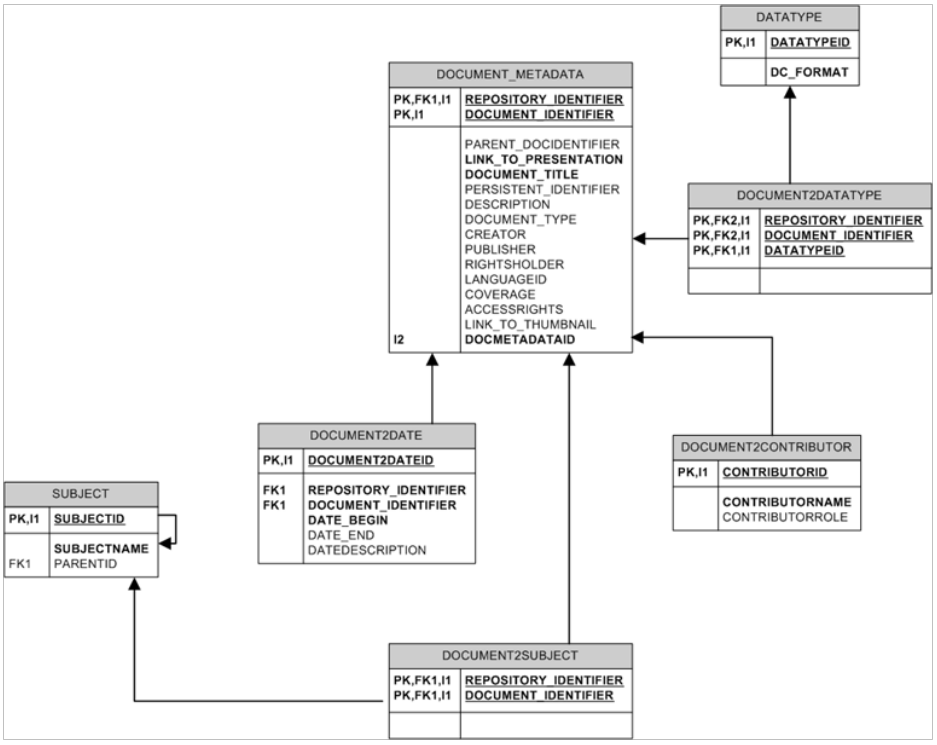


Abbildung 6.8: Metadatenmodell

- Erweiterung der Tabelle DOCUMENT_METADATA um zusätzliche Spalten wie PERSISTENT_IDENTIFIER (Repository-übergreifender eindeutiger Identifier) sowie urheberspezifische Informationen und Sprache. Hinzufügen eines künstlichen Primärschlüssels zur Unterstützung der Volltextsuche.
- Einem Dokument kann aktuell nur mehr eine einzige Sprache zugeordnet werden, daher entfällt die entsprechende Zuordnungstabelle und die Sprache wird in DOCUMENT_METADATA direkt abgelegt.
- Erweiterung der Tabelle DOCUMENT2DATE, um auch Zeiträume ablegen zu können.

6.4.4 Administratives Modell

Dieses Datenbankmodell dient zur Ablage von Benutzerdaten und findet sowohl im CORE als auch im 3D Repository Verwendung. Ursprünglich wurde es im Rahmen der dieser Diplomarbeit vorangegangenen Seminar-/Projektarbeit [Foi09] entwickelt und später angepasst, um zusätzliche Anforderungen des 3D Bereichs erfüllen zu können. Abbildung 6.9 stellt das administrative Datenmodell dar. Ziel des Datenmodells ist es, die unterschiedlichen Benutzertypen (Einzelpersonen, Firmen, Gruppenbenutzer) und Rollen abbilden zu können, sowie Zuordnungen von Benutzern zu Rollen vornehmen zu können. Die Rechtezuordnung kann entweder indirekt über Rollen oder direkt durch Zuordnung zu den einzelnen Benutzern erfolgen. Weiters können zusätzliche Daten wie Addressen, Orte und Städte verwaltet werden. Abbildung 6.11, entnommen aus [Foi09], gibt einen groben Überblick über den strukturellen Aufbau der Benutzer und Rechteverwaltung.

Die abgelegten Benutzerdaten können über eine Administrationskonsole verwaltet werden. Die Administrationskonsole wurde als Webservice mit einer Silverlight Oberfläche umgesetzt. In einer weiteren Ausbaustufe soll das Modell um eine Rechteverwaltung erweitert werden. Aktuell werden Benutzerdaten und Berechtigungen im Zuge von Suchanfragen noch außen vor gelassen.

Die in Abbildung 6.10 gezeigte Administrationskonsole ermöglicht es, über eine Oberfläche Personen, Benutzer und Rollen zu verwalten.

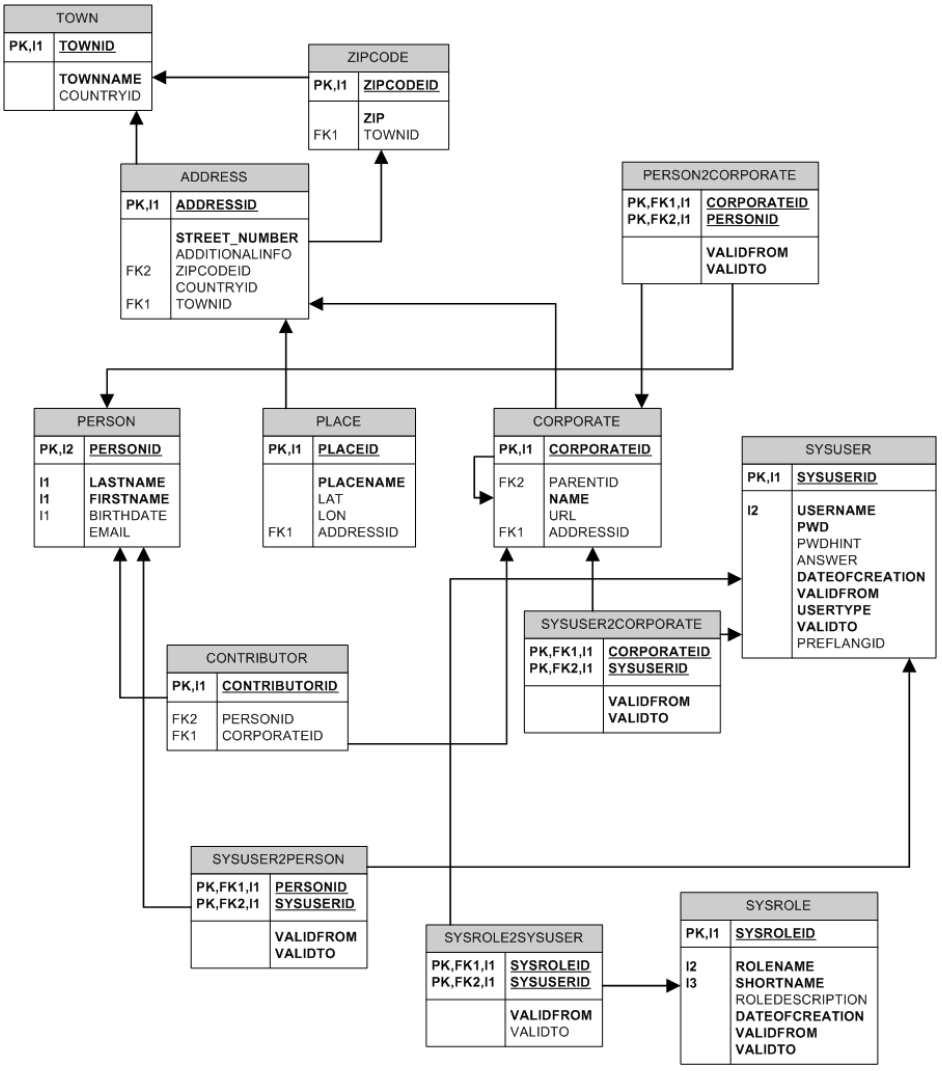


Abbildung 6.9: Administratives Datenbankmodel. Quelle: [Foi09]

6.4.5 Struktur zur Protokollierung von Suchabfragen

Die in Abbildung 6.12 gezeigte Datenstruktur wurde im Rahmen dieser Arbeit neu geschaffen und ermöglicht es, Suchabfragen, Suchergebnisse sowie Benutzerfeedback abzuspeichern. Die so gewonnenen Daten können einerseits für statistische Auswertungen verwendet werden (z.B. beliebteste Suchbegriffe), andererseits als Basis für ein personalisiertes Ranking von Suchergebnissen. Weiters bieten sie dem Benutzer die Möglichkeit, eine persönliche Suchhistorie inklusive Abfragen und Ergebnissen abzuspeichern.

Suchanfrage

Die Suchanfrage wird in den Tabellen SEARCHQUERY, SEARCHTERM und SEARCHORDER abgelegt. Resultate der Suchabfragen sowie Relevanzfeedback werden in der Tabelle QUERYRESULT gespeichert. Im Folgenden findet sich eine genaue Beschreibung der einzelnen Tabellen.

SEARCHQUERY

Allgemeine Daten zur Suchanfrage werden in der Tabelle SEARCHQUERY 6.2 gespeichert. Da die meisten dieser Daten aktuell noch nicht an das Service übermittelt werden, können die Spalten auch leer sein. Aktuell können das Datum der Suche, der Typ der abgefragten Objekte, das zu durchsuchende Re-

Abbildung 6.10: Oberfläche der Administrationskonsole. Quelle: [Foi09]

Spalte	Datentyp	Beschreibung
SEARCHDATE	DATETIME	Zeitpunkt der Suche
SESSIONID	INT	Session Identifier
USERID	INT	ID des angemeldeten Benutzers
IP_ADDRESS	NVARCHAR(30)	IP Adresse des Clients vom dem die Anfrage gesendet wird.
DOCUMENT_TYPE	INT	Typ der angefragten Dokumente (z.B. 3D)
REPOSITORY IDENTIFIER	NVARCHAR(128)	ID des zu durchsuchenden Repositories.

Tabelle 6.2: SEARCHQUERY

pository sowie benutzerspezifische Daten wie Session-ID, Benutzer-ID und IP-Adresse des verwendeten Clients abgelegt werden.

SEARCHTERM

Die Tabelle SEARCHTERM 6.3 wird zur Speicherung der mit einer Anfrage verknüpften Suchbegriffe verwendet. Hier werden die bei der Suche eingegebenen Suchkategorien und Suchbegriffe abgelegt.

SEARCHORDER

In der Tabelle SEARCHORDER 6.4 wird die gewünschte Reihung der Suchergebnisse abgelegt. Diese wird aktuell als durch Komma separierter String in der Spalte STMT_SEARCHORDER abgelegt. Es wurde eine eigene Tabelle gewählt, um ein Aufbrechen dieser Struktur in einer weiteren Ausbaustufe zu erleichtern.

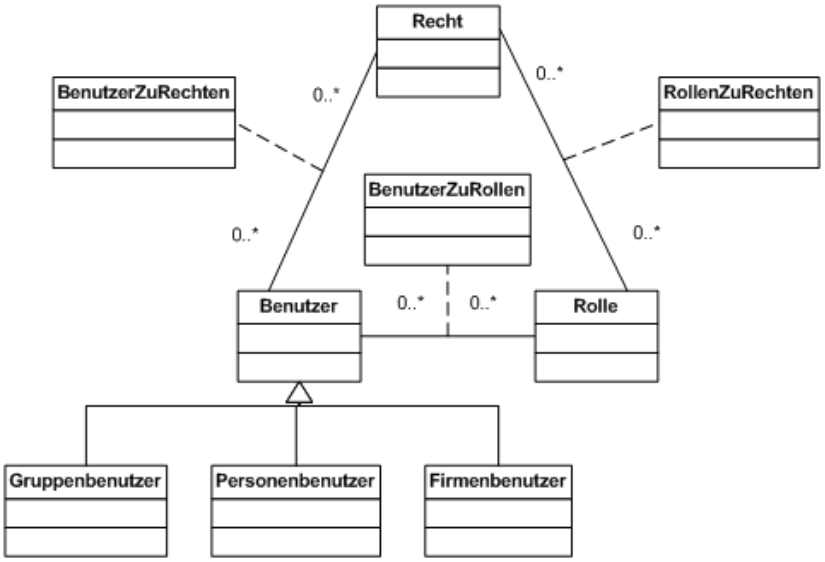


Abbildung 6.11: Struktureller Aufbau der Benutzer und Rechteverwaltung. Quelle: [Foi09]

Spalte	Datentyp	Beschreibung
SEARCHQUERYID	INT	Referenz auf die zugehörige Suchanfrage
SEARCHCATEGORY	NVARCHAR(256)	Metadatenkategorie oder Kennung für Volltextsuche
SEARCHVALUE	NVARCHAR(256)	Suchbegriff für Metadatensuche oder Volltextsuche

Tabelle 6.3: SEARCHTERM

QUERYRESULT

Die Tabelle QUERYRESULT 6.5 dient zur Protokollierung der zu einer Suchanfrage bestimmten Ergebnisse, sowie des zugehörigen Benutzerfeedbacks. Sie enthält detaillierte Informationen über Zusammensetzung und Reihung und Herkunft der Elemente der Ergebnisliste.

Protokollierung

Die Protokollierung der Suchanfragen ist defaultmäßig aktiviert. Sollte der Benutzer keine Protokollierung wünschen, kann er diese Option über das Suchinterface aktiv ausschalten. Zur Übermittlung dieser Option an das Service ist eine Erweiterung des WSDL-Protokolls notwendig.

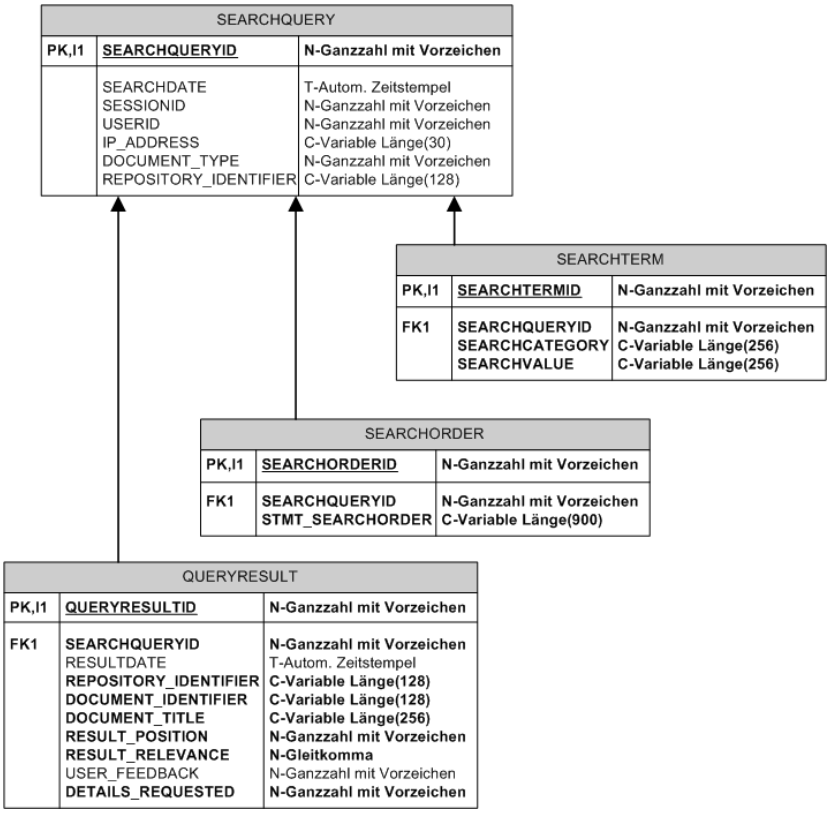


Abbildung 6.12: Struktur zur Protokollierung von Suchanfragen

Spalte	Datentyp	Beschreibung
SEARCHQUERYID	INT	Referenz auf die zugehörige Suchanfrage
STMT_SEARCHORDER	NVARCHAR(900)	Reihung der Suchergebnisse wird als durch Komma separierter String abgelegt.

Tabelle 6.4: SEARCHORDER

Spalte	Datentyp	Beschreibung
SEARCHQUERYID	INT	Referenz auf zugehörige Suchanfrage
RESULTDATE	DATETIME	Zeitpunkt der Bestimmung des Suchergebnisses
REPOSITORY_IDENTIFIER	NVARCHAR(128)	ID des Repositories aus dem das Suchergebnis stammt. (Ursprungsrepository des Dokuments)
DOCUMENT_IDENTIFIER	NVARCHAR(128)	Eindeutige Identifikation des Dokuments
DOCUMENT_TITLE	NVARCHAR(256)	Titel des Ergebnisdatensatzes (Dokuments)
RESULT_POSITION	INT	Position des Resultats in der Ergebnisliste
RESULT_RELEVANCE	FLOAT	Relevanzwert des Ergebnisses in Bezug auf die Suchanfrage
USER_FEEDBACK	INT	Vom Benutzer vergebener Relevanzwert
DETAILS_REQUESTED	INT	Wurde das Suchresultat vom Benutzer im Detail betrachtet (J/N)

Tabelle 6.5: QUERYRESULT

6.5 Personalisierung und Benutzerfeedback

Im folgenden Abschnitt soll die zunehmende Bedeutung von personalisierten Suchergebnissen sowie die Verbesserung der Treffsicherheit durch Benutzerfeedback näher untersucht werden.

6.5.1 Motivation

Bei der Suche in digitalen Bibliotheken ist die Qualität und Treffsicherheit der Suchergebnisse ein wesentlicher Faktor. Eine im Jahr 2008 von der GfK Marktforschung im Auftrag von Google durchgeführte repräsentative Studie über das Suchverhalten von Benutzern im Internet (suchmaschinenunabhängig) zeigte, dass nur 30% aller Benutzer mehr als drei Suchbegriffe verwendeten und der größte Teil der Benutzer (37%) sich sogar auf einen einzigen Suchbegriff beschränkten. Der Einsatz von Operatoren für eine gezieltere Suche wurde nur von 13% aller Teilnehmer genutzt. [Deu11] Das lässt den Schluss zu, dass sich bei Verwendung der Volltextsuche im Metadatenbestand, das Suchverhalten des durchschnittlichen Bibliotheksbenutzers nicht wesentlich von dieser Vorgehensweise abweicht. Trotzdem erwartet sich der Benutzer, anhand seiner Eingaben, genau die für ihn wichtigen Inhalte, nach Relevanz geordnet angeboten zu bekommen. In diesem Zusammenhang ist auch zu beachten, dass einerseits Suchbegriffe nicht immer eindeutig sind (Homonyme, Synonyme) und andererseits die Relevanz von Suchergebnissen durchaus benutzerspezifisch unterschiedlich beurteilt wird. Es ist also eine ständige, benutzerabhängige Verbesserung der eingesetzten Suchalgorithmen notwendig, um die Qualität der angebotenen Suchergebnisse zu verbessern. In diesem Zusammenhang spielen die Begriffe Benutzerfeedback und Personalisierung eine wichtige Rolle. Nur durch die Schaffung einer entsprechenden Hintergrundstruktur, die es ermöglicht Rückmeldungen zu Suchanfragen und Suchergebnissen auszuwerten, kann der Suchvorgang und somit auch die Relevanz der angebotenen Ergebnisse verbessert werden.

6.5.2 Benutzerfeedback

Hier unterscheidet man zwischen aktivem und automatischem Feedback. Bei ersterem bewertet der Benutzer aktiv die Relevanz der Suchergebnisse anhand einer Skala. Da diese Option für den Benutzer mit Arbeit verbunden ist, wird sie eher selten genutzt werden. Beim automatischen Feedback wird das Anklicken einzelner Resultate durch den Benutzer mitprotokolliert. Setzt z.B. ein Benutzer mehrmals eine Suchanfrage zum Thema „Auto“ ab und klickt dann jedesmal die Resultate zur Marke „VW“ an, werden diese bei weiteren Suchanfragen desselben Benutzers höher gereiht. Die Umsetzung beider Optionen erfordert eine Erweiterung des WSDL-Protokolls zur Übermittlung des Feedbacks an den CORE sowie eine Adaptierung des Frontends.

6.5.3 Personalisierung

Für ein personalisiertes Ranking der Suchergebnisse ist eine eindeutige Identifikation des Benutzers notwendig. Diese kann entweder über eine Anmeldung des Users am System Identifikation des Benutzers (Session ID, Cookie) erfolgen.

6.5.4 Statistische Auswertung

Im Rahmen einer statistischen Auswertung kann folgendes ermittelt werden:

- Bestimmung der beliebtesten Suchbegriffe
- Wie viele Suchbegriffe werden durchschnittlich eingegeben?

6.5.5 Protokollierung der Suchabfragen in PROBADO

Das Abspeichern der Suchanfragen, sowie der Suchergebnisse ist im Rahmen des Webservices bereits umgesetzt. Für einen weiteren Ausbau (Personalisierung, Feedback usw.) müssen zunächst die oben erwähnten Protokollerweiterungen, Service- und Frontendumbauten vorgenommen werden.

6.6 Webservices

Wie bereits in den vorangegangenen Abschnitten erwähnt erfolgt die Kommunikation zwischen den einzelnen Systemebenen durch Webservices über den Austausch von SOAP-Nachrichten. Dabei wurden zunächst die Schnittstellen mittels WSDL (Web Service Definition Language) exakt definiert. Diese WSDL-Dateien enthalten Informationen zu:

- Types: Fasst Übergabeparameter zu Klassen (Strukturen) zusammen.
- Messages: Beschreibt die verfügbaren Operationen.
- PortTypes: Definiert die Methoden Binding und Encoding: Bezeichnet das verwendete Protokoll – ist im Fall von PROBADO immer SOAP XML mit HTTP Version 1.2
- Service Endpoint

Im Zuge der Reimplementierung wurden die bestehenden WSDL Dokumente [Alt09] angepasst bzw. erweitert. Dies erfolgte in Zusammenarbeit mit der TU Darmstadt, wobei der Hauptteil der Arbeit dort durchgeführt wurde. Die aktuelle Version der Schnittstellenbeschreibungen befindet sich im Anhang.

Folgende Vorgänge wurden mittels WSDL spezifiziert:

- Registrierung eines neuen Repositories beim CORE (RepositoryRegistration.wsdl)
- Import von Metadaten in den CORE (RepositoryMetadataExchange.wsdl)
- API für Suchanfragen zwischen Frontend und CORE (ProbadoSearch.wsdl)
- API für Suchanfragen zwischen CORE und Spezialrepositorien (RepositorySearch.wsdl)

Schnittstellenübergreifend verwendete Datentypen wurden in ein eigenes XML-Dokument ausgelagert (ProbadoSharedSchema.xsd). Die WSDL Schnittstellenbeschreibungen bildeten die Basis für die Implementierung der CORE Webservices, die ebenfalls in Zusammenarbeit mit der TU Darmstadt vorgenommen wurde. Als Entwicklungsumgebung und Programmiersprache wurden Visual Studio 2008, sowie C# gewählt. Als Webserver wurde IIS (Internet Information Services) von Microsoft gewählt.

Bezeichnung	Beschreibung	Pflicht
Title	(string) Titel des Repositories	Ja
Identifier	(string) Eindeutige Kennung des Repositories	Ja
RepositorySearchSoapAddress	(string) Die Webadresse unter der das Repository seinen Such-Service anbietet	Ja
Description	(string) Optionale Beschreibung des Repositories.	Nein
Owner	(string) Optionale Information zum Besit-zer/Betreiber des Repositories.	Nein
TypesSupported	(enum, array) Vom Repository unterstützte Be-reiche – aktuell stehen die Typen META, 3D und MUSIC zur Verfügung.	Ja
Language	(enum, array) Unterstützte Sprachen des Repositories	Nein

Tabelle 6.6: Metadatensatz Repository Registrierung

6.7 Repository Registrierung

Um seiner Steuerungs- und Kontrollfunktion nachkommen zu können, benötigt der CORE gewisse Informationen über die angeschlossenen Repositorien. Diese Daten werden einerseits benötigt um sie dem Benutzer über das Frontend anzuzeigen (z.B. als Auswahlliste angeschlossener Repositorien oder zur Verfügung stehender Datentypen), andererseits um Anfragen weiterleiten und beantworten zu können. Der im Rahmen des Registrierungsvorganges ausgetauschte Metadatensatz beruht hauptsächlich auf Dublin-Core Elementen. Im Zuge des Registrierungsvorganges wird folgendermaßen vorgegangen:

1. Verpflichtend: Beantragung eines eindeutigen Repository Identifiers. Dieser Schritt erfolgt manuell.
2. Verpflichtend: Übermittlung eines Repository Metadatensatzes an den CORE.
3. Optional: Für jede im Repository zur Verfügung stehende Suchmaschine, die über das Frontend auswählbar sein soll, erfolgt die Übermittlung eines Suchmaschinen Metadatensatzes, der die Bezeichnung der Suchmaschine sowie die von ihr unterstützten Formate beinhaltet. Werden dem CORE hier keine Daten zur Verfügung gestellt, dann wird jede Suchanfrage, die an dieses Repository adressiert ist, ungeprüft weitergeleitet. Bei Vorhandensein der Daten, werden nur Suchanfragen, die sich auf vom Repository unterstützte Formate beziehen, weitergeleitet.

Zusätzliche Nachrichten zur Aktivierung/Deaktivierung einzelner Repositorien sowie zum Hinzufügen/Entfernen von Suchmaschinen stehen zur Verfügung.

6.7.1 Metadatensatz Repository Registrierung

Dieser Metadatensatz beinhaltet verpflichtend die (eindeutige) Bezeichnung des Repositories, dessen Kennung und Adresse sowie die von ihm unterstützten Typen (aktuell stehen hier Musik und 3D zur Auswahl). Weiters kann optional eine Beschreibung des Repositories, sowie eine Liste der unterstützen Sprachen übermittelt werden. Tabelle 6.6 gibt einen Überblick über die ausgetauschten Daten. [SAF10]

6.7.2 Metadatensatz Queryengine Registrierung

Jedes angeschlossene Repository verfügt über mindestens eine Suchmaschine, die bestimmte Abfrageformate unterstützt. Über den Metadatensatz “Queryengine Registrierung” können für alle angeschlossenen Suchmaschinen die Bezeichnung und die Liste der unterstützen Formate an PROBADO übermittelt werden. Tabelle 6.7 gibt einen Überblick über den bei der Registrierung von Queryengines verwendeten Metadatensatz.

6.7.3 Ablauf Repository Registrierung

Das Sequenzdiagramm 6.13 zeigt den Ablauf des Registrierungsvorganges. Zunächst muss für das anzuschließende Repository einmalig ein Identifier beantragt werden. Die Vergabe des Identifiers erfolgt

Bezeichnung	Beschreibung	Pflicht
EngineName	(string) Bezeichnung der Suchmaschine	Ja
QueryFormat	(string, array) Von der Suchmaschine unterstützte Abfrageformate – hier können sämtliche MIME-Typen sowie META für Metadatensuche und FULLTEXT für Volltextsuche angegeben werden.	Ja

Tabelle 6.7: Metadatensatz Queryengine Registrierung

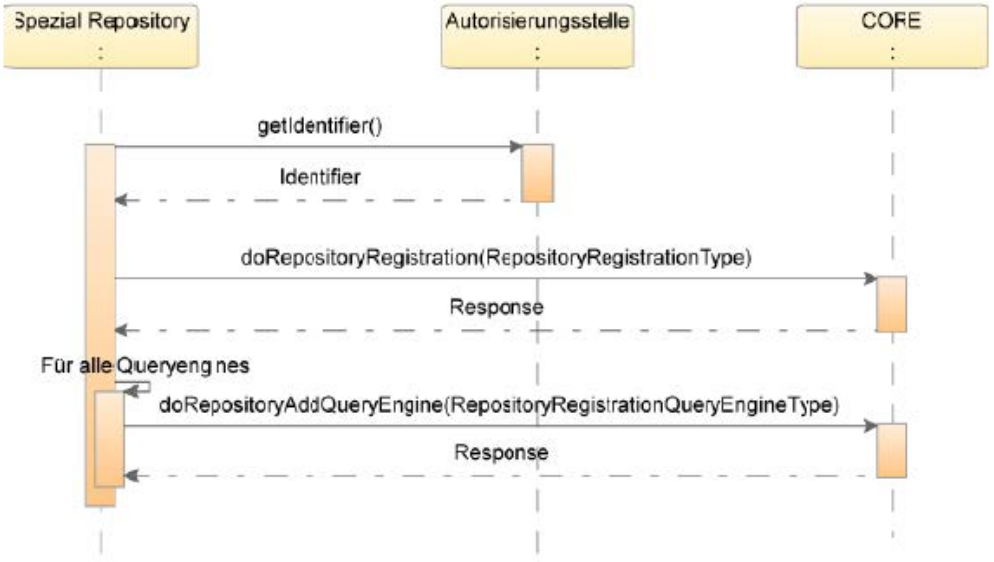


Abbildung 6.13: Zeitlicher Ablauf bei der Registrierung eines neuen Repository

nach Absprache mit dem Betreiber manuell. Nach Erhalt des Identifiers kann die Anmeldung des Repositorys und seiner Suchmaschinen bei PROBADO erfolgen. Dies erfolgt durch die Implementierung der SOAP-Aufrufe doRepositoryRegistration und doRepositoryAddQueryEngine innerhalb eines anhand der PROBADO Schnittstellenspezifikation geschaffenen Webservices durch das zu registrierende Repository.

6.7.4 SOAP Nachrichten Repository Registrierung

Im folgenden finden sich Beispielnachrichten, die während der Registrierung eines neuen Repositories ausgetauscht werden. Zunächst wird vom Spezial-Repository eine Registrierungsnachricht abgeschickt.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
      xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
      doRepositoryRegistration
  </Action>
  </s:Header>
  <s:Body>
    <RepositoryRegistration xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <Title>3D Repository</Title>
      <Identifier>3D_AXDFETHDS</Identifier>
      <RepositorySearchSoapAddress>http://MYSERVER:54320/myRepository
      </RepositorySearchSoapAddress>
      <Description>3D Testrepository</Description>
      <Owner>max</Owner>
```

```
      <TypesSupported xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <TypesSupportedByProbadoElement>META</TypesSupportedByProbadoElement>
        <TypesSupportedByProbadoElement>3D</TypesSupportedByProbadoElement>
      </TypesSupported>
      <Language xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <LanguageElement>GERMAN</LanguageElement>
      </Language>
    </RepositoryRegistration>
  </s:Body>
</s:Envelope>
```

Diese wird vom CORE empfangen und nach erfolgreicher Registrierung mit einer Bestätigung, andernfalls mit einer Fehlermeldung beantwortet.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <RepositoryRegistrationResponse xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <ResponseValue>true</ResponseValue>
    </RepositoryRegistrationResponse>
  </s:Body>
</s:Envelope>
```

Nach Erhalten der Antwortnachricht kann das Spezial-Repository optional die Registrierungsnachrichten für die einzelnen Queryengines schicken.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
      xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
     doRepositoryAddQueryEngine
  </Action>
  </s:Header>
  <s:Body>
    <RepositoryAddQueryEngine xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <RepositoryIdentifier>3D_AXDFETHDS</RepositoryIdentifier>
      <EngineName>QE_Metadata</EngineName>
      <QueryFormat xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <StringElement>META</StringElement>
      </QueryFormat>
    </RepositoryAddQueryEngine>
  </s:Body>
</s:Envelope>
```

6.7.5 Änderung von Repository Daten

Ein einmal registriertes Repository bleibt in der Datenbank enthalten und muss bei Bedarf manuell gelöscht werden. Im Falle von Datenänderungen schickt das Spezial-Repository erneut einen Registrierungsdatensatz an den CORE, der prüft ob das Repository bereits vorhanden ist und,wenn ja, ein Update auf die bestehenden Daten durchführt.

Abbildung 6.14: Zeitlicher Ablauf beim Austausch der Metadaten

6.8 Metadatenaustausch

Der nächste (optionale) Schritt nach der Registrierung eines Spezial-Repository ist der Export eines Metadaten-Subsets an den CORE. Da dieser Vorgang in der ersten Implementierung zwar geplant, aber noch nicht umgesetzt war, wurde hier gemeinsam mit der TU Darmstadt eine neue WSDL Spezifikation und aufbauend auf dieser ein Webservice zum Import der Metadaten in den CORE erstellt. Dieses Service ermöglicht es den angeschlossenen Spezial-Repositorien, einen ausgewählten Teil ihrer Metadaten per PUSH an den CORE zu übermitteln. Diese Daten werden im CORE gespeichert und für die lokale Metadatensuche verwendet, können jedoch nicht geändert werden. Datenänderungen erfolgen ausschließlich in den Spezial- Repositorien. Änderungen können über einen erneuten, vom Herkunfts-Repository ausgelösten Export in den CORE übernommen werden.

6.8.1 Vorgehen beim Export

Über die Methode doRepositoryMetadataExchange(RepositoryMetaDataExchangeType) wird ein Metadatensatz an den CORE übermittelt. Dieser prüft, ob der Datensatz bereits vorhanden ist und führt in Abhängigkeit davon ein Update oder Insert durch. Tabelle 6.8 (entnommen aus der CORE Dokumentation) beinhaltet sämtliche Parameter, die über den RepositoryMetaDataExchangeType übergeben werden können. [SAF10] Abbildung 6.14 zeigt den zeitlichen Ablauf des Metadatenimports in den CORE.

Im Anschluss findet sich noch eine Beispielnachricht für den Import von Metadaten in den CORE.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
     xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
     doRepositoryMetadataExchange</Action>
  </s:Header>
  <s:Body>
    <RepositoryMetadataExchange xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <RepositoryIdentifier>3D_AXDFETHDS</RepositoryIdentifier>
      <DocumentIdentifier>MY_DOC1</DocumentIdentifier>
      <ParentDocumentIdentifier />
      <LinkToPresentation>ftp://test.dyns.org/modelriegers.jpg
      </LinkToPresentation>
      <PersistentIdentifier>doi</PersistentIdentifier>
      <Title>Riegersburg</Title>
      <Description>Die Riegersburg erhebt sich auf dem Felsen eines Vulkanberges..
      </Description>
      <TypeOfDocument>_3D</TypeOfDocument>
      <Contributor xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <ContributorElement>
```

```
            <Name>Max Mustermann</Name>
            <Role>Archivar</Role>
          </ContributorElement>
        </Contributor>
        <Creator>Elisabeth Katharina von Galler</Creator>
        <AccessRights>FULL</AccessRights>
        <Format xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
          <StringElement>META</StringElement>
        </Format>
        <Subject xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
          <StringElement>Architektur</StringElement>
          <StringElement>Burg</StringElement>
        </Subject>
        <Language>GERMAN</Language>
        <Publisher>Testfirma</Publisher>
        <Rightsholder>Testfirma</Rightsholder>
        <Date xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
          <DateTypeElement>
            <Description>Enstehung</Description>
            <DateBegin>1141-01-01T00:00:00</DateBegin>
            <DateEnd>1150-01-01T00:00:00</DateEnd>
          </DateTypeElement>
        </Date>
        <Coverage>Steiermärkische Landesaustellung</Coverage>
       <LinkToThumbnail>http://test.dyns.org/riegers.jpg<LinkToThumbnail/>
      </RepositoryMetadataExchange>
    </s:Body>
</s:Envelope>
```

Parameter	Beschreibung	Pflicht
RepositoryIdentifier	Eindeutiger Identifier des Repositories. (string)	Ja
DocumentIdentifier	Eindeutiger Bezeichner für das Dokument innerhalb des Repositories. Bildet gemeinsam mit dem Repository-Identifier den eindeutigen Schlüssel für das Dokument. (string)	Ja
ParentDocumentIdentifier	Identifier des übergeordneten Dokuments. (string)	Nein
LinkToPresentation	Link zu einer Präsentation des Werkes bzw. Aufruf-parameter für ein Anzeige-Applet. Der Inhalt der Vorabpräsentation wird vom jeweiligen Repository präsentiert und gehostet. (string)	Ja
PersistentIdentifier	DOI/URN des Dokuments. (string)	Nein
Title	Titel/Bezeichnung des Dokuments. (string)	Ja
TypeOfDocument	Typ des Dokuments. Zur Zeit werden von PROBADO die Typen 3D, MUSIC und META unterstützt. (enum)	Ja
Description	Textuelle Beschreibung des Dokuments. (string)	Nein
Contributor	Liste von Personen, Firmen, Gruppen, usw., die einen Beitrag zu diesem Dokument geleistet haben. Komplexer Typ bestehend aus Name (string) und Rolle (string)	Nein
Creator	Person, Gruppe, Firma usw., die hauptverantwortlich für die Entstehung dieses Dokuments ist. (string)	Nein
Publisher	Person, Gruppe, Firma usw. die hauptverantwortlich für die Veröffentlichung dieses Dokuments ist. (string)	Nein
RightsHolder	Rechteinhaber des Dokuments. (string)	Nein
Format	Format des zum Dokument gehörenden Datenobjekts (z.B. MIMETYPE). (string)	Ja
Subject	Liste von Kategorien, denen das Dokument zuordenbar ist. (string)	Nein
Date	Liste von signifikanten Zeitpunkten (Zeitspannen), die mit dem Dokument in Verbindung stehen. Kom-plexer Typ, enthält Datumsbeschreibung (string), Startzeit-punkt (date) und Endzeitpunkt (date).	Nein
Coverage	Raum und Zeit der von diesem Dokument über-spannt wird. (Z.B. „frühe Renaissance 1420-1500 Italien“.	Nein
LinkToThumbnail	Link zu öffentlich zugänglichem Thumbnail des Werkes.	Nein
BinaryThumbnail	Binäre Bilddaten eines Thumbnails.	Nein
Language	Sprache des Dokuments. (enum, aktuell zur Verfügung stehende Werte sind GERMAN, ENGLISH, UNKNOWN = Defaultwert)	Nein
AccessRights	Zugriffsberechtigungen	Nein

Tabelle 6.8: Parameter Repository Metadatenexport

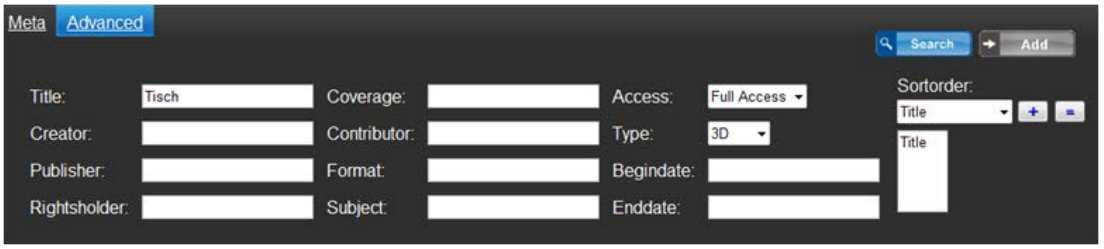


Abbildung 6.15: Eingabe der Suchanfrage und der gewünschten Ergebnisreihung ins Suchinterface

6.9 Suchanfragen

Da dieser Punkt bereits in der Diplomarbeit von Monika Alter ausführlich behandelt wurde, soll in diesem Abschnitt nur ein kurzer Überblick über die Suche an sich gegeben werden, sowie auf die im Zuge der Reimplementierung durchgeführten Änderungen und Anpassungen näher eingegangen werden.

6.9.1 Allgemein

Wie bereits zu Beginn dieses Kapitels erwähnt, gliedert sich die Suche in zwei Bereiche: in Anfragen die vom Frontend an den CORE gerichtet (ProbadoSearch) sind und in Anfragen, die vom CORE an die jeweiligen Spezialrepositorien weitergeleitet werden (RepositorySearch).

6.9.2 ProbadoSearch

Dieser Service bildet die definierte externe Schnittstelle zur Suchfunktionalität des Systems. Sämtliche Suchanfragen seien sie an den CORE oder an ein/mehre Spezial Repositorien gerichtet, müssen über diesen Service gestellt werden. Die an PROBADO gestellten Suchanfragen lassen sich grundsätzlich in zwei Typen unterteilen: Metadatensuche und inhaltsbasierte Suche. Der grundsätzliche Ablauf ist dabei immer derselbe:

- Frontend: Eingabe der Abfrage durch den Benutzer
- Zusammenbau und Versenden einer SOAP-Nachricht an den CORE
- Empfang und Dekodierung der Nachricht im CORE
- Durchführen einer SQL-Abfrage auf die CORE-Metadaten
- Erstellen einer Ergebnisliste.
- Verpacken der Ergebnisliste in eine SOAP-Nachricht.
- Versenden der SOAP-Nachricht an das Frontend.

Der beschriebene Ablauf wird von Grafik 6.15 dargestellt.

Die SOAP Nachricht im Anschluß ist ein Beispiel für eine vom Client generierte Nachricht, die an den Server versandt wird.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1"
     xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">
     doProbadoMetadataSearch
    </Action>
  </s:Header>
  <s:Body>
    <ProbadoMetadataSearch xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <Metadata xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <KeyValuePairElement>
        <Key>TITLE</Key>
```

```
          <Value>Tisch</Value>
          <Specification i:nil="true" />
        </KeyValuePairElement>
        <KeyValuePairElement>
          <Key>DOCUMENTTYPE</Key>
          <Value>ITEM_3D</Value>
          <Specification i:nil="true" />
        </KeyValuePairElement>
        <KeyValuePairElement>
          <Key>ACCESS</Key>
          <Value>FULL</Value>
          <Specification i:nil="true" />
        </KeyValuePairElement>
      </Metadata>
      <SortBy xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <KeyValuePairElement>
          <Key>TITLE</Key>
          <Value>asc</Value>
          <Specification i:nil="true" />
        </KeyValuePairElement>
      </SortBy>
      <CoreSearch>true</CoreSearch>
      <RepositoryId xmlns:i="http://www.w3.org/2001/XMLSchema-instance"/>
      <StartIndex>0</StartIndex>
      <Count>11</Count>
      <SessionId />
    </ProbadoMetadataSearch>
  </s:Body>
</s:Envelope>
```

Der CORE empfängt die SOAP Nachricht, decodiert sie und baut dynamisch eine SQL-Abfrage zusammen, die auf der CORE-Datenbank abgefragt wird. Das Ergebnis dieser Abfrage wird in eine Ergebnisliste verpackt und als SOAP-Nachricht an das Frontend verschickt. Dieser Ablauf wird von Abbildung 6.16 bildlich dargestellt. Die Ergebnisliste wird in eine SOAP-Nachricht verpackt und an den anfragenden Client versandt. Die SOAP Nachricht im Anschluß stellt ein Beispiel hierfür dar.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <ProbadoMetadataSearchResponse xmlns="http://www.probado.de/wsdl/ProbadoSchema">
      <TotalResultsCount>10</TotalResultsCount>
      <ResultElements xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <SearchResultElementArrayElement>
          <RepositoryIdentifier>MY_REP1</RepositoryIdentifier>
          <DocumentIdentifier>5</DocumentIdentifier>
          <Ranking>1</Ranking>
          <Position i:nil="true" />
          <Title> 230 Aline_Tische</Title>
          <AccessRights>FULL</AccessRights>
          <TypeOfDocument>_3D</TypeOfDocument>
          <LinkToPresentation>http://mimas.cgv.tugraz.at/Repository3D/
            ModelDetails.aspx?id=5
          </LinkToPresentation>
          <LinkToThumbnail i:nil="true" />
          <AdditionalBinaryData i:nil="true" />
        </SearchResultElementArrayElement>
            </ResultElements>
      <StartIndex>0</StartIndex>
      <Count>10</Count><SessionId />
    </ProbadoMetadataSearchResponse>
  </s:Body>
</s:Envelope>
```

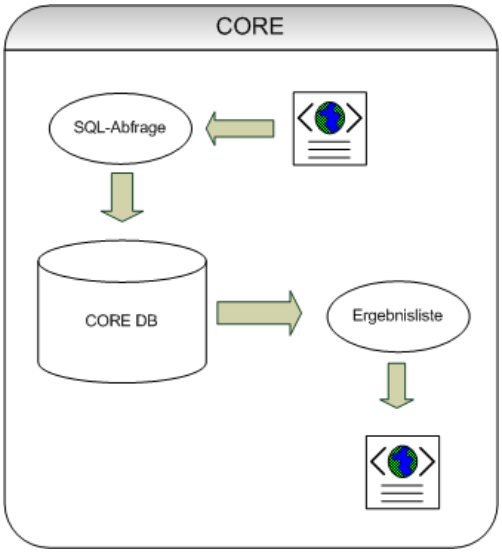


Abbildung 6.16: Abfrage im CORE

Abbildung 6.17: Darstellung der Ergebnisliste im Frontend

Diese Nachricht wird an das Frontend übermittelt, decodiert und dargestellt, wie in Abbildung 6.17 zu sehen.

Der Unterschied zwischen Metadaten und inhaltsbasierter Suche liegt einerseits in der Struktur der Abfrageparameter, andererseits in der Art und Weise, wie die Abfrage im CORE durchgeführt wird.

Abfrageparameter Metadatensuche

Die möglichen Abfrageparameter der Metadatensuche werden in der WSDL-Spezifikation mit dem Datentyp ProbadoMetadataSearchType festgelegt. Unter inhaltsbasierten Metadaten sind beispielsweise ASCII kodierte Feature-Vektoren für RUG oder globale Shape-Deskriptoren zu verstehen. Dieser Datentyp wurde ursprünglich von Monika Alter spezifiziert [Alt09] und im Zuge der Reimplementierung angepasst. Tabelle 6.9 gibt einen Überblick über die vorhandenen Attribute.

Parameter	Beschreibung	Pflicht
MetaData	(Komplexer Datentyp) Liste der Key (Metadatenfeld) / Value (Suchbegriff) Paare nach denen gesucht werden soll.	Ja
SortBy	(Komplexer Datentyp) Liste der Key (Metadatenfeld) / Value (Sortierrichtung) nach denen die Ergebnisliste sortiert wer-den soll.	Ja
CoreSearch	(bool) Ist dieses Flag gesetzt, wird die Anfrage nur im CORE ausgeführt.	Ja
RepositoryId	(string) Liste eindeutiger Identifier der Repositorien in denen gesucht werden soll. (Default = CORE) Für den Fall, dass das Flag CoreSearch nicht gesetzt ist, leitet der CORE die Anfrage an alle Repositorien, die den angefragten Datentyp unterstützen, weiter.	Ja
StartIndex	(int) Anzahl der zu überspringenden Resultate. Wird für das Weiterblättern in Ergebnislisten benötigt.	Ja
Count	(int) Anzahl der von Frontend angeforderten Resultate. (Default = 10). Ist CORE-seitig auf einen Maximalwert von 100 beschränkt.	Ja
SessionId	Client erhält bei der ersten Anfrage eine Session ID, die er bei künftigen Anfragen mitgeben muss.	Ja

Tabelle 6.9: Abfrageparameter Metadatensuche

Parameter	Beschreibung	Pflicht
MetaDate	(Komplexer Datentyp) Liste der Key (Metadatenfeld) / Value (Suchbegriff) Paare nach denen gesucht werden soll.	Ja
SortBy	(Komplexer Datentyp) Liste der Key (Metadatenfeld) / Value (Sortierrichtung) nach denen die Ergebnisliste sortiert wer-den soll.	Ja

Tabelle 6.10: Abfrageparameter inhaltsbasierte Suche

Abfrageparameter Inhaltsbasierte Suche

Die Abfrageparameter 6.10 für die inhaltsbasierte Suche werden über den Datentyp ProbadoSearchContentType in der WSDL Spezifikation festgelegt. Sie sind abgesehen von den ersten drei Attributen ident mit den Attributen von ProbadoMetadataSearchType.

6.9.3 CORE Metadatensuche

Dabei handelt es sich um eine Abfrage spezieller Metadaten über SQL-Statements. Es kann so eine spezifische Suche auf einzelne Felder wie z.B. Titel, Urheber, Rechteinhaber oder den den Entstehungszeitraum durchgeführt werden. Tabelle 6.11 listet die bei der Metadatensuche berücksichtigten Felder im Detail auf.

Das Suchstatement wird aus Performancegründen unter Umgehung der LINQ2SQL Syntax harcoded dynamisch zusammengebaut und ausgeführt. Sämtliche vom Benutzer eingegebenen Einschränkungen sind mit UND verknüpft.

Ranking Metadatensuche

Die Suchabfrage für die Metadatensuche ist UND-verknüpft. Das heißt alle zurückgelieferten Ergebnisse entsprechen den Bedingungen (haben im Vergleich zur Volltextsuche also einen Ranking Wert von 1000). Das Ranking, umgesetzt auf die Reihung der Ergebnisliste, ergibt sich implizit durch die gewählte Sortierreihenfolge der Resultate.

6.9.4 CORE Inhaltsbasierte Suche

Unter einer inhaltsbasierten Suche ist im CORE-Bereich immer eine Volltextsuche zu verstehen. Für die technische Umsetzung wurde dazu die Volltextsuche von MS SQL-Server eingesetzt. Hierzu musste zunächst das Volltextsuchfeature von SQL-Server 2008 aktiviert werden und ein Volltextindex für alle zu durchsuchenden Tabellen erstellt werden. Dieser Volltextindex ist sprachabhängig, das heißt er muss

Feld	Beschreibung	Suchmodus
TITLE	Titel des Dokuments	LIKE
DOCUMENTTYPE	Dokumententyp (3D oder Musik)	EQUALS
CREATOR	Urheber	LIKE
PUBLISHER	Veröffentlicht von	LIKE
RIGHTSHOLDER	Rechteinhaber	LIKE
LANGUAGE	Sprache des Dokuments (**)	EQUALS
COVERAGE	Thema, Event	LIKE
ACCESS	Zugriff (Voll oder Kein Zugriff)	EQUALS
FORMAT	Format in dem das Doku-ment vorliegt. (*)	EQUALS
SUBJECT	Thema, Schlagwort	LIKE
BEGINDATE	Beginndatum	>=
ENDDATE	Endedatum	<=
CONTRIBUTOR	Beitragender	LIKE
ROLE	Rolle des Beitragenden	LIKE

Tabelle 6.11: Datenbankfelder Metadatensuche

Tabelle	Felder	Volltextprädikat
DOCUMENT_METADATA	Titel, Description	FREETEXT
DATATYPE	Format	FREETEXT
DOCUMENT2CONTRIBUTOR	Contributorname, Contributorrole	FREETEXT
SUBJECT	Subjectname	FREETEXT
DOCUMENT2DATE	Datedescription	FREETEXT

Tabelle 6.12: Datenbankfelder Volltextsuche

für jede zu unterstützende Sprache erzeugt werden. Aktuell wird nur die deutsche Sprache unterstützt. Die in Tabelle 6.12 angeführten Felder werden bei der Volltextsuche berücksichtigt:

Ranking Volltextsuche

Bei der Volltextsuche wird vom SQL-Server in der Spalte RANK ein Integer Ranking Wert vergeben, der zwischen 1 und 1000 liegt. Je höher das Ranking desto besser entspricht das Resultat der Suchanfrage. Ranking Werte beziehen sich immer auf eine spezifische Suchanfrage, daher sind abfrageübergreifende Vergleiche nicht möglich.

6.9.5 Ergänzende Schnittstellenfunktionen CORE Suche

Um zusätzliche Funktionen, wie Browsing in vorhandenen Suchbegriffen, die Suche in einem Subject-Baum, oder die Verwendung eines sogenannten „Query Bags“ für kombinierte Suchanfragen abzudecken, wurden dem WSDL-Protokoll noch folgende Schnittstellenfunktionen hinzugefügt.

- doProbadoGetCoreMetadata
- doProbadoQueryBagSearch
- doProbadoTreeBrowse

Da diese Funktionen aktuell nur für die Suche in den Spezial-Repositorien im Einsatz sind, werden sie hier nicht näher beschrieben.

6.10 RepositorySearch

Der CORE kann bei Bedarf/Verlangen Suchanfragen direkt an die angeschlossenen Repositorien weiterleiten, deren Ergebnisse er dann entsprechend ihrer Reihung zusammenfasst und an das Frontend weiterleitet. Dazu müssen die externen Repositorien die vom CORE spezifizierten Schnittstellenfunktionen doRepositoryContentSearch und doRepositoryMetadataSearch implementieren und bei der Registrierung des jeweiligen Repositorys eine Adresse bekanntgeben, unter der dieser Webservice konsumiert werden kann. Die tatsächliche Implementierung der Suche ist dem externen Repository überlassen, und kann unterschiedlich ausfallen. Weiters stehen noch zusätzliche Schnittstellenfunktionen, vergleichbar den in Punkt 6.8.5. beschriebenen zur Verfügung, um die Möglichkeit zu schaffen, Anzeigedaten für das Suchinterface über den CORE ans Frontend weiterzuleiten.

Kapitel 7

Datenbankanpassungen im 3D Bereich

In diesem Kapitel wird ein Überblick über die im Rahmen dieser Arbeit erbrachten praktischen Arbeiten in bezug auf die PROBADO 3D Datenbank gegeben. Diese umfassten die folgenden Punkte:

- Wechsel von MYSQL zu SQL-Server
- Neugestaltung des Datenbankmodells
- Übernahme und Konvertierung bestehender Daten.

Die in diesem Kapitel beschriebenen Aufgaben wurden im Rahmen dieser Arbeit in Absprache mit dem 3D-Bereich in Graz und Bonn sowie der TIB-Hannover durchgeführt.

7.1 Gestaltung eines neuen Datenbankmodells

Im Zuge des Technologiewechsels in der zweiten Projektphase wurde beschlossen, sowohl den CORE als auch den 3D-Bereich datenbanktechnisch auf MS SQL-Server umzustellen. Zugleich sollte das bestehende Datenbankmodell grundsätzlich überarbeitet und an die aktuellen Bedürfnisse des 3D Bereichs angepasst werden.

Abbildung 7.1 gibt einen Überblick über das überarbeitete 3D Datenbankschema.

7.1.1 Schwachstellen des ursprünglichen Datenbankmodells

Das ursprüngliche Datenbankmodell war im Zuge der ersten Projektphase stückweise entstanden. Immer wenn neue Funktionalitäten hinzukamen und somit ergänzende Daten abzulegen waren, wurde es Zug um Zug erweitert. Das führte dazu, dass viele Daten redundant abgelegt waren und auf die optimale Gestaltung eines relationalen Datenbankmodells (Normalformen) vielfach keine Rücksicht genommen wurde. Auch wurde das Datenbankdesign sehr stark an das Applikationsdesign angelehnt, sodass der Inhalt einzelner Tabellen Klassen in der Applikation reflektierte. Dies führte zu Redundanzen und in weiterer Folge zu Dateninkonsistenzen sowie Performanceeinbußen. Zusätzlichen Handlungsbedarf sah man in der Ablage mehrsprachiger Texte, der generisch gestaltet werden sollte.

7.2 Vorgangsweise

Folgende Vorgangsweise wurde umgesetzt:

- Installation von MS-SQL-Server Express Edition und Aufsetzen einer MS SQL-Server Datenbankinstanz.
- Überarbeiten des Datenbank-Modells
- Einmaliger Export bestehender Daten aus der alten MySQL-Instanz und Import in die MS SQL-Server Instanz.

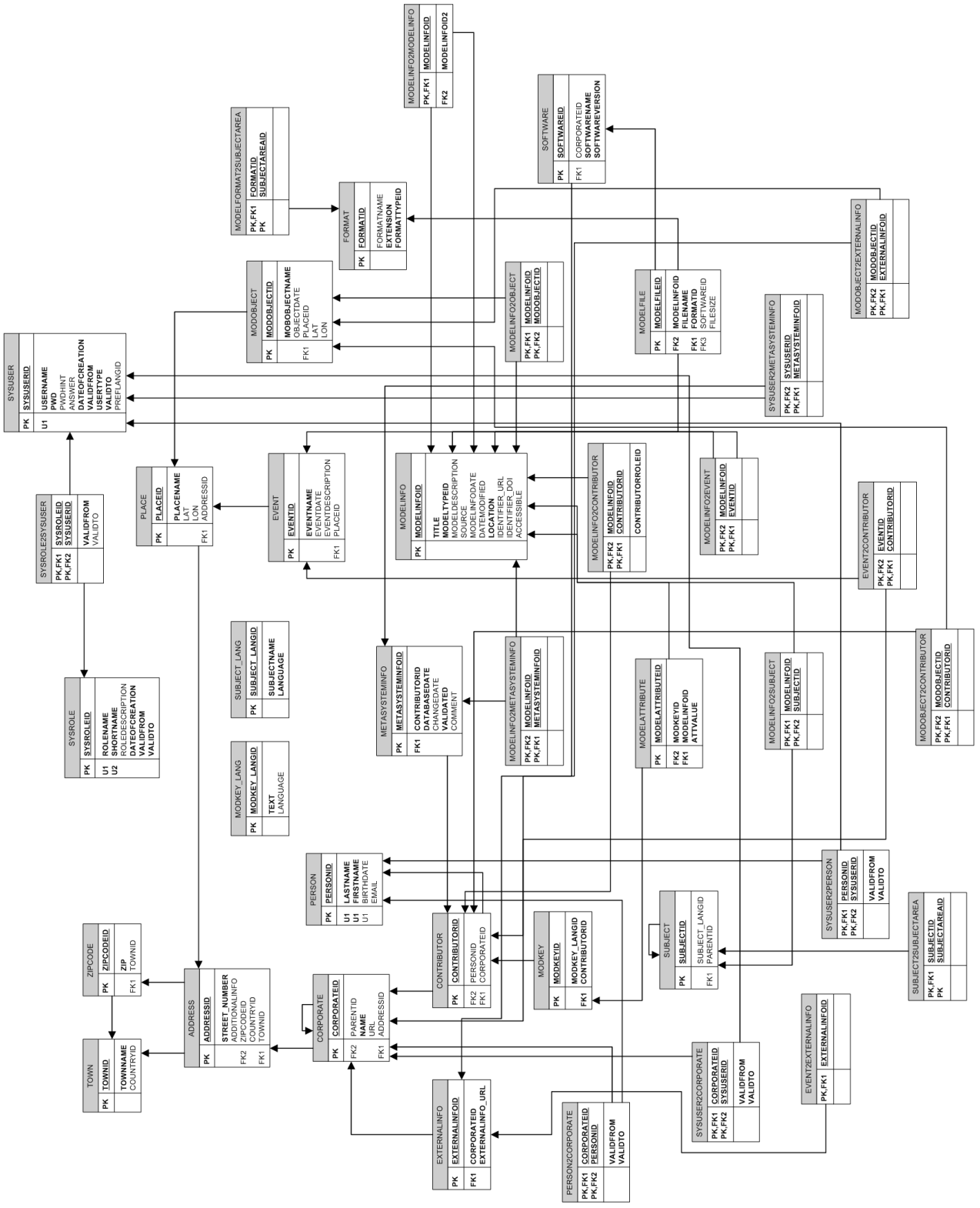


Abbildung 7.1: Überarbeites 3D Datenbankmodell

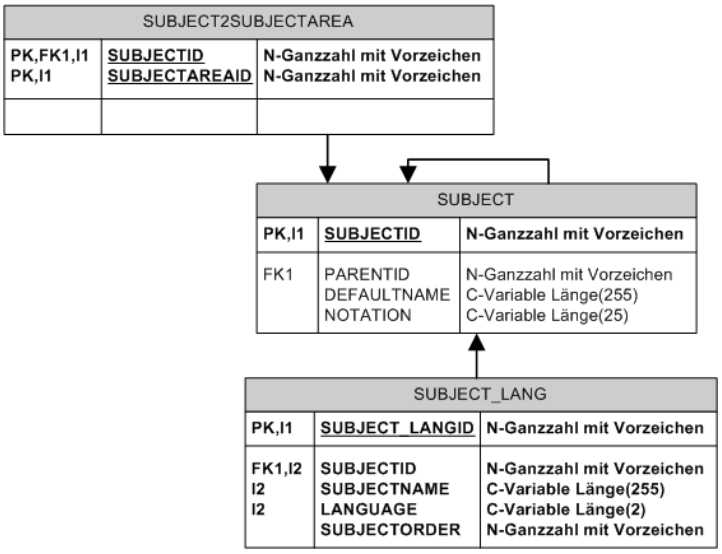


Abbildung 7.2: Tabellenstruktur zur Ablage von Schlagwörtern

7.2.1 Überarbeitung Datenbankmodell

Im Zuge der Überarbeitung des Datenbankmodells wurden die anschließenden Punkte umgesetzt:

- Benennung der Tabellen und Spalten anhand einer einheitlichen Namenskonvention.
- Durchgängige Vergabe von Primär- und Fremdschlüsseln für alle Tabellen, sowie Indizierung wichtiger Spalten.
- Auslagerung redundanter Daten in eigene Tabellen.
- Austausch vorhandener Benutzertabellen, durch die Tabellenstruktur der CORE-Benutzer und Rechteverwaltung.
- Auslagerung von Enumerationsdaten ins Listenmodell (vgl. CORE)
- Umbau einiger Stammdatentabellen zur Unterstützung mehrerer Sprachen.
- Schaffung einer Reihe von Views und Prozeduren für einen vereinfachten applikationsseitigen Zugriff auf bestimmte Daten.
- Umfassende Dokumentation der Datenbankstruktur.

7.3 Mehrsprachigkeit

Ein wesentliches Ziel bei der Restrukturierung des Datenbankmodells war die Unterstützung verschiedener Sprachen. Es sollte einerseits auf einfache Weise möglich sein, verschiedensprachige Interfaces, mit der Datenbank zu verbinden, und z.B. Stammdaten wie unterstütze Suchformate oder Schlagworte in der gewählten Sprache anzeigen zu können. Auch Volltextsuche für nicht deutschsprachige Inhalte sollte unterstützt werden.

7.3.1 Umsetzung im Datenbankschema

Im Rahmen des Datenbankschemas wurde die Unterstützung von mehrsprachigen Texten einerseits im Rahmen des Modells für Enumerationsdaten (siehe Kapitel 6) gewährleistet, andererseits durch den Einbau einer ergänzenden Tabellenstruktur.

Das hier angewandte Vorgehen soll am Beispiel der Tabellenstruktur zum Ablegen von Schlagworten (siehe 7.2) näher erklärt werden.

Die in dieser Tabellenstruktur abzulegenden Schlagwörter sollten dem Benutzer im Webinterface in Form eines Schlagwortbaumes angezeigt werden. Ändert der Benutzer im Webinterface die eingestellte Sprache von Deutsch auf Englisch, wird auch der Schlagwortbaum in Englisch dargestellt. Der Aufbau der Baumstruktur erfolgt über die Tabelle SUBJECT. Hier wird die Struktur, sowie eine Default-Bezeichnung (für den Fall, dass für einen Eintrag keine Übersetzung vorhanden ist), eingetragen. In der zugehörigen Tabelle SUBJECT_LANG ist im Feld SUBJECTNAME die Bezeichnung für die jeweilige Sprache eingetragen, die über ein Kürzel im Feld LANGUAGE definiert wird. Über das Feld ORDER kann die Anzeigereihung von Synonymen gesteuert werden. Als Basis für die Übersetzung dient dabei ein deutscher Schlagwortbaum, dieser bildet die Grundstruktur, die z.B. in ein Excel-Sheet exportiert, übersetzt und mittels Script wieder eingespielt werden kann. Der Vorteil dieses Modells liegt im relativ einfach gestaltbaren Übersetzungsworkflow. Ein negativer Aspekt ist die Bindung an den strukturellen Aufbau des Basisbaums.

7.3.2 Volltextsuche

Um auf Datenbankebene eine sprachabhängige Suche durchführen zu können, sind mehrere Faktoren zu berücksichtigen. Standard SQL erlaubt zwar die Suche nach Textbestandteilen (Operator LIKE, Wildcards), nimmt aber keine Rücksicht auf Abwandlungen, Synonyme oder abweichende, aber phonetisch ähnlich klingende Schreibweisen eines Begriffs. Um auch diese Faktoren einzubeziehen, muss eine Volltextsuche eingesetzt werden. Grundvoraussetzung sind zunächst einmal die Verwendung von Unicode als Zeichensatz, sowie das Vorhandensein einer entsprechenden Search-Engine, die es erlaubt einen sogenannten Volltextindex für Textfelder anzulegen. Diese Features werden heutzutage von sämtlichen gängigen DBMS zumindest in Form eines Zusatzmoduls zur Verfügung gestellt. Allerdings ist der Einsatz in den meisten Fällen auf eine einzige (wählbare) Sprache beschränkt. So ist auch bei MS SQL-Server nur ein einziger Volltextindex pro Tabelle zulässig. Um also innerhalb einer Datenbank eine Volltextsuche für verschiedene Sprachen umsetzten zu können, muss entweder für jede Sprache eine eigene Tabelle oder ein eigener indizierter View angelegt werden [SR07]. Die Volltextsuche im 3D Bereich wurde vom Entwicklungsteam des 3D Bereichs an der TU Graz realisiert. Aktuell wird sie nur für die deutsche Sprache unterstützt.

7.3.3 Export und Import bestehender Daten

Da das Datenbankschema der Zieldatenbank sich beträchtlich vom Schema der Quelldatenbank unterschied, wurde der Ex- und Import mit Hilfe einer Kombination von SQL-Skripten und PLSQL-Prozeduren durchgeführt, die die Daten einerseits ergänzten, andererseits auf die entsprechenden Zieltabellen verteilten.

Kapitel 8

Benutzerinterface für mobile Endgeräte

"If it doesn't exist on the internet, it doesn't exist" - Kenneth Goldsmith wies bereits im Jahre 2005 in seinem Statement auf die zunehmende Bedeutung des Internets als Informationsmedium und des sich verändernden Zugangs zu Informationen hin. Nicht nur die Art und Weise, wie Information gespeichert wird, auch die Suche nach Informationen hat im letzen Jahrzehnt einen radikalen Wandel erfahren. Physikalische Medien wie Bücher, Zeitschriften oder CDs werden in zunehmendem Ausmaß von digitalen Medien wie eBooks, elektronischen Zeitungsausgaben, oder Musikdateien abgelöst. Parallel dazu hat sich auch das Suchverhalten radikal geändert. Selbst für eine akademische Recherche wird in vielen Fällen zunächst Google[1] als Medium der Wahl herangezogen und erst später, basierend auf diesen Suchergebnissen, wenn überhaupt, eine Suche in einer Literaturdatenbank durchgeführt [Gol12].

Eine weitere Ursache dieses Wandels, ist die zunehmende Verbreitung mobiler Endgeräte wie Smartphones oder Tablets, die dem Benutzer zusätzlich zur klassischen Telefonie immer mehr Funktionen, die in der Vergangenheit vollwertigen Rechnern vorbehalten waren, anbieten. Eine 2007 von Lee Rainie, Jenna Anderson und anderen durchgeführte Studie belegt, dass mobile Endgeräte bis zum Jahre 2020 bei weitem das primäre Medium für den Zugriff auf das Internet darstellen werden [RA08]. Ein Trend, der sich in den Entwicklungen der letzten beiden Jahre wiederspiegelt, wobei die Veröffentlichung des ersten iPADs durch Apple, sowie der Markteintritt von Google in den Smartphonesektor und die zunehmende Verbreitung von Android wesentliche Meilensteine darstellten. Im Zuge dieser Entwicklung wird in zunehmendem Maße auf die speziellen Erfordernisse dieser Endgeräte eingegangen, die einerseits mehr Möglichkeiten als traditionelle Rechner bieten (z.B. GPS, Portabilität, Telefonie), andererseits aber auch Beschränkungen unterliegen (Bildschirmgröße, Rechenleistung, Betriebssystem, usw.). Mobile Informationsgewinnung ist somit als Spezialgebiet der klassischen Informationsgewinnung zu verstehen.

Forschungsansätze in diesem Bereich konzentrieren sich auf zwei Themengebiete: Kontext (*"context awareness"*) und inhaltliche Anpassung (*"content adaption"*). Dabei versteht man unter ersterem den Output des Gerätes an den Benutzer unter Berücksichtigung zusätzlicher Informationen wie z.B. dem aktuellen Standort des Benutzers und den sich daraus ergebenden Möglichkeiten, unter lezterem die Anpassung/Optimierung des Inhalts für die Darstellung auf einem mobilen Endgerät [TEX*10]. Abbildung 8.1, entnommen aus [TEX*10] gibt einen Überblick über die Themengebiete der mobilen Informationsgewinnung.

Um diesem Trend zu mobilen Endgeräten Rechnung zu tragen, wurde beschlossen PROBADO 3D, nach Abschluss der letzten Projektphase, noch um mobile Applikationen zu ergänzen. Dabei wurde zunächst versucht, einen Überblick über die aktuellen technischen Möglichkeiten zur Anbindung der existierenden SOAP Schnittstelle, sowie zur Darstellung von 3D Objekten auf mobilen Endgeräten zu gewinnen. Basierend auf den gewonnenen Erkenntnissen und dem bestehenden System entschloss man sich zunächst für Implementierungen für die Betriebssysteme Windows Mobile und iOS. Im Folgenden wird dieser Entscheidungsprozess näher erläutert. Das Projekt wurde bereits vor der Fertigstellung von Windows Phone 8 begonnen wurde, weshalb Windows Phone 8 im Zuge der folgenden Analyse nicht berücksichtigt wurde.

[1] www.google.com

Abbildung 8.1: Teilgebiete der mobilen Informationsgewinnung [TEX*10]

8.1 SOAP Unterstützung auf mobilen Plattformen

Der Begriff SOAP-Unterstützung wird oft unterschiedlich interpretiert. Häufig wird bereits die Möglichkeit, XML-Dokumente zu parsen und zu verarbeiten, als SOAP-Unterstützung bezeichnet. Diese reine XML-Verarbeitung ist jedoch für semantisch komplexe Schnittstellen ungeeignet, da sie die Vorteile einer semantischen Beschreibung der Nachrichten nicht nützt, und jede Änderung der Schnittstelle eine manuelle Anpassung auf Client-Seite nach sich zieht. Ein weiteres Merkmal für SOAP-Unterstützung ist die Möglichkeit, Datentypen und Client-Stubs zu einer vorhandenen WSDL-Beschreibung einer Schnittstelle, automatisch zu generieren. Dies vereinfacht einerseits die Implementation, und ermöglicht es andererseits, geringfügige Änderungen an der Schnittstelle sowie Änderungen des Web-Servers weitgehend automatisiert nachzuziehen. Ebenfalls wünschenswert ist die Unterstützung beliebiger, in XML-Schema definierter, Datentypen, um so eine Nutzung zusätzlicher, in SOAP integrierter Spezifikationen (z.B. für Sicherheitsfunktionen) zu ermöglichen. [ES12]

Tabelle 8.1 , entnommen aus [ES12] gibt einen Überblick über die SOAP Unterstützung auf aktuellen mobilen Plattformen:

Wie in der Gegenüberstellung deutlich erkennbar, ist die SOAP-Unterstützung der aktuellen mobilen Plattformen in vielen Bereichen stark eingeschränkt [ES12]. Aufgrund ihrer Verbreitung wurden für PROBADO nur drei der in Tabelle 8.1 angeführten Plattformen (Android, Windows Phone 7 sowie iOS) ernsthaft in Betracht gezogen.

8.1.1 Android

Bei Android, das aktuell in der Version 4.2 vorliegt [And13], handelt es sich um ein von Google und einer Gruppe von Hardwareherstellern entwickeltes, linux-basiertes Betriebssystem. Unterstützt von Firmen wie Motorola, Samsung und HTC hat es sich zum populärsten Betriebssystem für mobile Endgeräte entwickelt. [IGN12] Für die Entwicklung von Anwendungen kommt eine java-ähnliche API zum Einsatz, die die Verarbeitung von XML-Dokumenten unterstützt. Für SOAP-Webservices ist keine native Unterstützung vorhanden, diese muss über eine externe Zusatzbibliothek (z.B. kSOAP2) erfolgen. Ein Nachteil dieser Bibliothek ist, dass die Abbildung der Datentypen in Klassen nicht automatisiert erfolgt, was den Implementierungsaufwand erhöht. [ES12]

8.1.2 Windows Phone 7

Windows Phone 7 (WP7) wurde von Microsoft im Herbst 2010 als Nachfolger von Windows Mobile 6 veröffentlicht. Es bietet mit der Unterstützung von Silverlight als Plattform für Anwendungsentwicklung die Möglichkeit bestehende .NET Anwendungen und Bibliotheken für WP7 einzusetzen. Bibliotheken für die Unterstützung der Windows Comminuction Foundation sowie volle XML-Unterstützung ist vorhanden. Die Anbindung an Webservices wird durch die automatische Generierung von Client Stubs für Visual Studio vereinfacht. [ES12]

Plattform	API	SOAP nativ	SOAP mit Add-On
Android	Java SE/Java ME ähnlich	XML ja, Rest nur manuell	Stub-Generierung möglich, Datentypen manuell
Windows Mobile 6	.NET	XML und Generierung Datentypen und Stubs, aber nur WS-I Basic Profile 1.1 nutzbar	Für WS-I Basic Profile 1.1 nicht erforderlich, sonst gSOAP möglich
Windows Phone 7	.NET	XML und Generierung Datentypen und Stubs, aber nur WS-I Basic Profile 1.1 nutzbar	Nicht möglich
iOS	Objective C	Kaum: XML ja, WSDL stark eingeschränkt, sonst manuell	gSOAP siehe letzte Zeile
Blackberry OS	Java ME ähnlich	XML ja, JSR-172 [10] ja, sonst manuell	Stubgenerierung möglich, Datentypen manuell
Andere C/C++ basierte Plattformen	C/C++	-	sehr gut (mit gSOAP Aufsatz)

Tabelle 8.1: Überblick über die SOAP Unterstützung auf aktuellen mobilen Plattformen [ES12]

8.1.3 iOS

iOS wurde von Apple als Plattform für die Erstellung von Anwendungen für iPad/iPhone in Objective-C unter Verwendung der Cocoa API, angepasst auf die Bedürfnisse mobiler Geräte, entwickelt. Es bietet keine native Unterstützung für SOAP, sondern ausschliesslich für die Verarbeitung von XML-Nachrichten. Folgende externe Lösungen stehen zur Verfügung [ES12]:

- *wsdl2obj*:
  erzeugt einen Objective-C Client Stub aus der WSDL Schnittstellenbeschreibung, bietet aber nur teilweise Unterstützung des WSDL Standards. [ES12]

- *gSOAP*:
  dabei handelt es sich um eine kommerzielle Lösung, die alle genannten Anforderungen unterstützt und da auf ANSI C/C++ basiert auch für iOS einsetzbar ist. Das Lizenzmodell ist flexibel, es steht zusätzlich zur kommerziellen Lizenz auch eine GPL Lizenz zur Verfügung. Nachteilig ist, dass es bei jeder Betriebssystemanpassung manuell portiert werden muss. [ES12]

- *sudzc*:
  bezeichnet ein unter der Apache 2.0 Lizenz stehendes Projekt, das basierend auf einer WSDL-Schnittstellenbeschreibung serverseitig durch Einsatz von :NET/C# und XSLT die benötigten iOS Artefakte und Datenstrukturen für Remote Calls erzeugt. [Sch12]

8.2 Darstellung von 3D Objekten auf mobilen Endgeräten

Ein weiterer wichtiger Punkt war die Möglichkeit, die in PROBADO 3D gespeicherten 3D Modelle auch im Rahmen einer Vorschau auf mobilen Endgeräten anzeigen zu können. Diese Modelle werden in der 3D Datenbank zwar in verschiedenen Formaten abgelegt, aber im Rahmen des bestehenden Silverlight-basierten Webservices erfolgte die Darstellung in der Vorschau durch Download eines PDF 3D, das anschliessend mit Acrobat Reader angezeigt wird.

PDF Version 1.6 und höher (Acrobat Reader größer gleich Version 7.0) ermöglichen neben der Darstellung von Texten und Grafiken auch die Einbettung von 3D Inhalten in PDF Dokumente. [Aus12]

Abbildung 8.2: PROBADO 3D Detailansicht [pro11]

Die Abbildungen 8.2 und 8.3 der PROBADO3D Beta-Version veranschaulichen diesen Vorgang. Über den Link “3D Vollbild anzeigen” wird das in der Datenbank hinterlegte PDF 3D heruntergeladen und in einem 3D-fähigen PDF-Viewer angezeigt.

Die Anforderungen für die Anzeige der 3D Vorschau auf mobilen Engeräten sind weniger umfangreich und umfassen nicht das gesamte Funktionspektrum des 3D PDF-Viewers.

Folgende Aktionen sollen möglich sein:

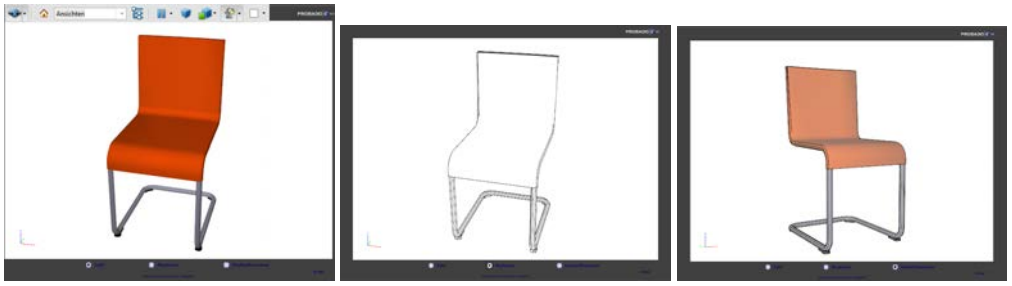
- Anzeige des Modells ohne Textur mit Standardbeleuchtung
- Drehen
- Verschieben
- Verkleinern/Vergrößern

Da zum Zeitpunkt der Entstehung dieser Arbeit weder für iOS, Android noch für Windows Phone 7 ein PDF Viewer existiert, der die die Anzeige von 3D-Objekten unterstützt, musste auf alternative Anzeigemöglichkeiten zurückgegriffen werden. Im Zuge der Auswahl wurden zunächst X3DOM, OpenGL und DirectX betrachtet.

8.2.1 X3DOM

Bei X3DOM handelt es sich um eine Initiative des Fraunhofer IGD gemeinsam mit dem dem Web 3D Konsortium mit dem Ziel interaktiven 3D Inhalt direkt in den HTML-5 Standard einzubetten. Durch die direkte Integration des X3D-Graphen in den HTML DOM enfällt die Notwendigkeit der Integration spezieller APIs oder Browser-Plugins. [IGD12] X3DOM stellt eine komplette plattformunabhängige 3D Render-Engine dar, aufbauend auf dem WebGL Standard, einem Java Script Interface zu OpenGL ES. [AH11]

In bezug auf mobile Browser werden HTML5 und im besonderen X3DOM aktuell nur von Firefox Mobile für Android unterstützt. [X3D12]



(a) Solid (b) Illustration (c) Shaded Illustration

Abbildung 8.3: Anzeigearten PDF 3D [pro11]

8.2.2 OpenGL

Seit seiner ersten Veröffentlichung 1992 durch das OpenGL Architecture Review Board (ARB) hat sich OpenGL zur am weitesten verbreiteten 2D und 3D Grafik API entwickelt. Seine Weiterentwicklung wurde vom ARB gesteuert, das 2006 in die Open GL Arbeitsgruppe unter der Schirmherrschaft des Khronos Konsortiums umgewandelt wurde. OpenGL wird von allen wichtigen Betriebssystemen und den meisten Programmiersprachen unterstützt. [Ope12] Es stellt eine hardwarunabhängige Schnittstelle zur Grafikhardware mit mehr als 700 Funktionsaufrufen zur Verfügung, über die komplexe Grafikobjekte aus einfachen geometrischen Primitven (Punkte, Linien, Dreiecke, Polygone) aufgebaut werden können. [Cha08] Mit OpenGL ES (Open GL for Embedded Systems), erstmalig 2003 auf der SIGGRAPH vorgestellt [Rid10b], steht eine Plattform für mobile Endgeräte zur Verfügung, die sowohl von Android als auch von iOS unterstützt wird, jedoch nicht von WP7, hier ist die einzig mögliche Option der Einsatz von DirectX mit XNA. [ZZZY11] OpenGL ES kommt in den Versionen 1.1 und 2.0 zum Einsatz.

8.2.3 NinevehGL

Bei Nineveh GL handelt es sich um eine Objective-C geschriebene 3D Engine auf Basis von Open GL ES 2.0. Sie ermöglicht es, sämtliche Features der programmierbaren Pipeline von OpenGL ES 2.0 zu nutzen, während zugleich der Import von Wavefront obj oder Collada Files ohne zusätzliche APIs möglich ist.

Nineveh GL ist Freeware und steht zur Zeit in der Beta-Version 0.9.2 zum Test bereit. [Bom12]

8.2.4 DirectX

DirectX, ursprünglich von Microsoft als High-Performance Plattform zur Spieleentwicklung entworfen, hat sich mittlerweile zu einem fundamentalen Teil von Windows entwickelt und wird als Teil der Windows SDK ausgeliefert. [Mic12a] Es handelt sich dabei um eine Reihe von APIs zum direkten Zugriff auf Grafik- Sound und Eingabegeräte sowie Netzwerkschnittstellen auf Windows Plattformen. DirectX besteht aus fünf Haupt APIs (Direct3D, DirectPlay, DirectSound, DirectInput und DirectDraw) wobei Direct3D für die Darstellung von 3D Objekten sowohl im Spiele bereich (Desktop oder XBox) als auch für CAD/CAM Anwendungen eingesetzt wird. [Mic12b]

8.2.5 Vergleich der untersuchten APIs

Da X3DOM aktuell nur von Firefox Mobile unterstützt wird und man nicht auf die Vorteile einer nativen App (u.a. Look & Feel, Bedienbarkeit, Geschwindigkeit) verzichten wollte, wurden in weiterer Folge nur OpenGL ES, NinevehGL und DirectX gegenübergestellt.

Tabelle 8.2 gibt einen Überblick über die im vorangegangenen Abschnitt beschriebenen APIs und deren Unterstützung durch die einzelnen Plattformen. Es zeigt sich hier, dass sämtliche untersuchten APIs mit Ausnahme von OpenGL, das sowohl von iOS als auch von Android unterstützt wird, plattformspezifisch sind. Dies bedeutet, dass zumindest zwei voneinander unabhängige Implementierungen notwendig sind, wobei eine Umsetzung für Windows nur mit DirectX möglich ist.

Plattform	OpenGL ES 2.0	NinevehGL	DirectX
Android	✔		
Windows Phone 7			✔
iOS	✔	✔	

Tabelle 8.2: Überblick über die 3D Unterstützung auf aktuellen mobilen Plattformen

8.3 Zusammenfassung

Schlussendlich waren für die Umsetzung einer mobilen Lösung folgende Faktoren ausschlaggebend:

- Wiederverwendung bestehender Codebestandteile
- SOAP Unterstützung
- 3D Darstellung
- Anwenderpräferenzen (z.B. besondere Affinität der Zielgruppe für einen Gerätehersteller)

Da das vorhandene Webservice als Windows Applikation auf Basis von Silverlight und SQL-Server umgesetzt wurde, WP7 eine vollständige SOAP Unterstützung bietet und die mobile Applikation somit in der gewohnten Entwicklungsumgebung unter Wiederverwendung bestehender Applikationsteile (mit Ausnahme der 3D Darstellung) umgesetzt werden kommte, war WP7 zunächst die logische Wahl. Als zusätzliche Plattform wurde iOS gewählt. Ausschlaggebend war hier die Präferenz der Anwender (Architekten, Grafiker, Designer) für Apple Produkte und die Nachfrage nach entsprechenden Apps.

8.4 Probado 3D für Windows Phone 7

Bei der aktuellen Lösung für Windows Mobile 8.4 handelt es sich um eine WP7-App basierend auf Silverlight, die von René Berndt im Rahmen des PROBADO Projektes entwickelt wurde. Für die Darstellung der 3D-Modelle wird DirectX eingesetzt. Der Aufbau und die Bedienung der App sind dem Webservice nachempfunden. Der Benutzer hat die Möglichkeit, einen Suchbegriff einzugeben, mit dem eine Volltextsuche in der Probado 3D Datenbank durchgeführt wird. Das Suchergebnis wird in Form einer scrollbaren Ergebnisliste mit Vorschaubild (siehe Abbildung 8.5) angezeigt. Durch das Anklicken eines Listeneintrages wird eine scrollbare Detailanzeige 8.5 geöffnet, die technische Daten, eine detaillierte Beschreibung des Objektes, sowie einen Link zum Öffnen einer 3D Vorschau enthält. Bei der 3D Detailanzeige handelt es sich um einen interaktiven Viewer, der es dem Benutzer ermöglicht, das Modell zu drehen, zu verschieben, sowie die Größe zu ändern. Es kann zwischen den Darstellungsarten “Wireframe” und “Animation” gewechselt werden. Beispiele dafür sind in den Abbildungen 8.6 zu sehen.

8.5 iProbado 3d

iProbado 3D 8.7 ist eine App für iPad/iPhone die es ermöglicht die PROBADO 3D Datenbank zu durchsuchen, Suchergebnisse im Detail zu betrachten, sowie wenn vorhanden, eine Vorschau des 3D Modells anzuzeigen. Sie wurde im Rahmen dieser Arbeit in Objective-C für den Einsatz mit iOS 5.0 und höher entwickelt. Wie die im vorigen Abschnitt beschriebene WP7 App ist auch iProbado 3D in bezug auf Aufbau und Bedienung an das PROBADO 3D Webservice angelehnt. Für die 3D Darstellung wird OpenGL ES 2.0 verwendet, unter Einsatz des seit iOS 5.0 von Apple zur Verfügung gestellten GLKit Frameworks [App13].

iProbado 3D ist eine Universal App, deren Oberfläche sowohl für die Verwendung auf dem iPad als auch auf dem iPhone optimiert wurde. Die Bedienung erfolgt analog zur WP7 App. Nach Eingabe eines Suchbegriffs in das Suchfeld wird eine Suchanfrage an das Probado 3D Webservice gestellt und die ersten 10 Resultate der Volltextsuche in einer Ergebnisliste dargestellt (siehe Abbildungen 8.8 und 8.9). Jeweils weitere 10 Resultate können durch Klicken auf den Link “weitere anzeigen” nachgeladen werden. Die Auswahl eines Resultates führt zur Anzeige einer Detailansicht (zu sehen in den Abbildungen

Abbildung 8.4: Probado 3D App für WP7 [Ber12]

8.8, 8.10 und 8.9), die ein größeres Vorschaubild, eine genaue Beschreibung und zusätzliche Informationen wie technische Details, Rechteinhaber, Events usw. enthält. Die Detailansicht ist einerseits zur Homepage des Bereitstellers des 3D Objektes verlinkt , andererseits kann über den Button 3D Vorschau das Modell in einem 3D Viewer angezeigt werden. 8.11.

8.5.1 Ablauf der Abfragen

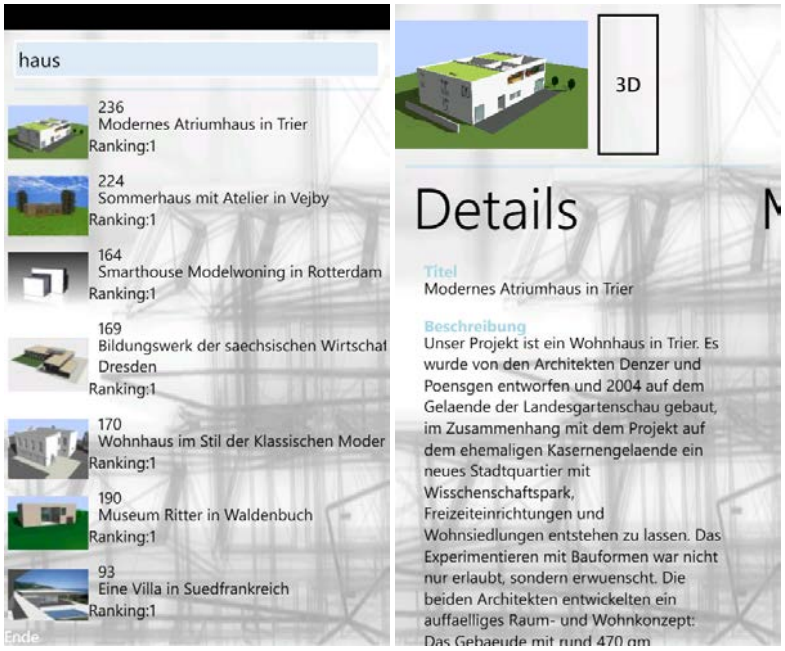
Suchanfrage Nach Eingabe eines Suchbegriffs durch den Benutzer wird zunächst ein get-Request mit dem Suchbegriff als Parameter an das PROBADO Webservice geschickt. Die Antwortnachricht mit den Suchergebnissen wird geparst und die erhaltenen Suchergebnisse werden in einer internen Ergebnisliste gespeichert, wobei die ersten zehn Ergebnisse sofort ausgegeben werden. Für jedes der dargestellten Ergebnisse wird sofort asynchron ein post-Request zur Abfrage des zugehörigen Vorschaubildes abgesetzt, welches nach Erhalt der Antwortnachricht nachgeladen wird. Abbildung 8.13 stellt diese Vorgangsweise in Form eines Sequenzdiagramms dar. Während des Nachladens der Vorschaubilder ist der Button "Weitere anzeigen" deaktiviert. Erst nach Abschluss dieses Vorgangs wird er wieder aktiviert und der Benutzer kann weitere zehn Ergebnisse aus der internen Ergebnisliste anzeigen lassen.

Detailanzeige Das Anklicken eines Elementes der Ergebnisliste durch den Benutzer führt zum Absetzen eines post-Requests zur Abfrage der Modelldetails an das PROBADO Webservice.

3D Vorschau Das Modell für die 3D Vorschau wird über einen get-Request vom PROBADO Webservice angefordert und in Form einer in der Antwortnachricht verpackten obj Datei übermittelt. Diese wird aus der Nachricht extrahiert, eingelesen und in der 3D Vorschau mittels OpenGL dargestellt.

8.5.2 3D Darstellung in iOS

3D Rendering, Geometrie Unter Rendering (Erzeugen eines zweidimensionalen Bildes aus 3D Daten) versteht man die Darstellung von Objekten auf zweidimensionalen Oberflächen in einer Art und Weise, dass sie vom Auge des Betrachters als dreidimensional wahrgenommen werden. Das resultierende Bild ist aus Pixeln, bestehend aus RGB-Werten zusammengesetzt. Um Grafiken möglichst schnell und effizient anzuzeigen, besitzen moderne eingebettete Systeme einen graphischen Subprozessor (GPU) , dessen Aufgabe es ist, geometrische Daten, Farben, Beleuchtung und andere Informationen, die benötigt werden um ein Bild auf dem Bildschirm darzustellen, zu kombinieren. [Buc12]



(a) Ergebnisliste Probado 3D WP7 (b) Ergebnisliste Probado 3D WP7

Abbildung 8.5: Ergebnisliste und Detailansicht Probado 3D WP7 [Ber12]

OpenGL Bei OpenGL handelt es sich nur um eine von mehreren Methoden der 3D-Darstellung die von iPhone/iPad unterstützt werden. Die von iPhone/iPad verwendeten APIs lassen sich (wie von Abbildung 8.14 dargestellt) grob in vier Ebenen einteilen: Cocoa Touch, Media Services, Core Services und Core OS. Die beiden untersten Ebenen Core Services und Core Operation Systems (auch als *Darwin* bezeichnet) sind dieselben wie in MacOS, das Handling in den darüberliegenden Ebenen unterscheidet sich jedoch grundlegend, da iOS nicht wie MacOS OpenGL in seiner Gesamtheit sondern das schlankere OpenGL ES unterstützt.

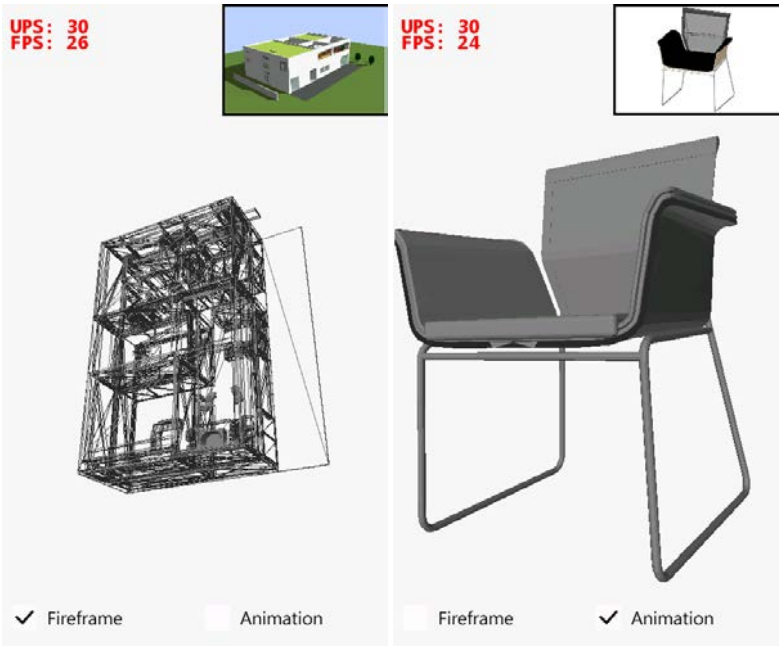
Die eingesetzte Grafik-Technologie besteht aus folgenden Bestandteilen:

- Quartz 2D Render Engine: leistungsstarke vektorbasierte Grafikbibliothek zum Rendern von Text und Grafik.
- Core Graphics: Vanilla C Interface für Quartz.
- UIKit: Natives Fenster-Framework für Quartz. Bietet einen Objective-C Wrapper für Quartz.
- Cocoa Touch: Konzeptioneller Layer für iPhone/iPad , der neben UIKit noch weitere Frameworks enthält.
- Core Animation: Objective-C Framework für komplexe Animationen
- Open GL ES: low level grafikbasierte C API zum Rendern von 2D und 3D Grafiken
- EAGL: API als Schnittstelle zwischen OpenGL ES und UIKit. Ein Teil der Klassen ist im Quartz Core Framework definiert, ein Teil im OpenGL ES Framework.

OpenGL ES wird in der Version 1.1 von sämtlichen iPhone/iPod Versionen unterstützt, die 2007 veröffentlichte Version 2.0 jedoch nur iPhone Modellen ab iPhone 3GS, sowie iPad, da erst diese Geräte über eine programmierbare Grafik-Pipeline verfügen.

[Rid10a]

Der wesentliche Unterschied zwischen den Versionen darin besteht, dass die Version 1.1 eine fixe Pipeline verwendet (Beleuchtung, Kanten, Farben, Kamera usw. werden über Funktionsaufrufe gesteuert), während bei OpenGL ES 2.0 eine programmierbare Pipeline zum Einsatz kommt. Anstelle der Funktionsaufrufe treten vom Programmierer erstellte Shader, was einerseits zu einem Mehraufwand bei der Codierung führt, andereseits aber zu enorme Vorteile in bezug auf Performance und Flexibilität mit



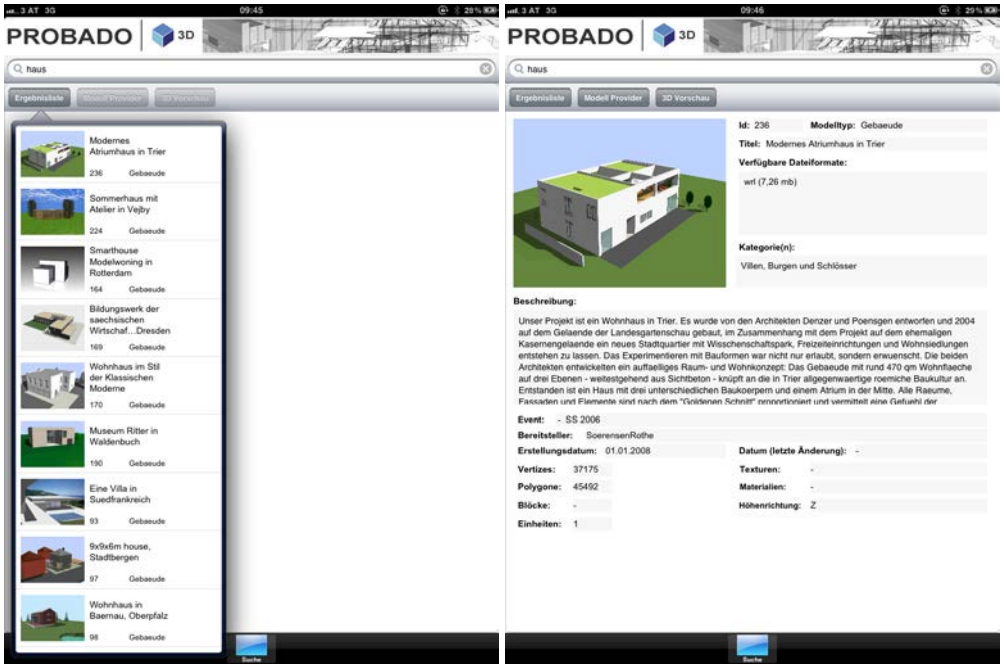
(a) 3D Vorschau Wireframe Probado 3D WP7

(b) 3D Vorschau Animation Probado 3D WP7

Abbildung 8.6: 3D VorschauProbado 3D WP7 [Ber12]

Abbildung 8.7: iProbado 3d

sich bringt. [Wen12] Die Pipelines werden durch die Abbildungen 8.15 und 8.16, entnommen aus der OpenGL ES Spezifikation [Ope12] dargestellt. Seit iOS 5.0 stellt Apple mit GLKit ein Framework zur Verfügung, dass die fehlende Infrastruktur im Vergleich zu Open GL ES 1.1 ersetzt und die Programmierung vereinfacht [Buc12].



(a) Ergebnisliste iPad (b) Detailansicht iPad

Abbildung 8.8: Ergebnisliste und Detailansicht iPad

(a) Ergebnisliste iPhone (b) Detailansicht iPhone (c) Detailansicht iPhone

Abbildung 8.9: Ergebnisliste, Detailansicht und 3D Darstellung iPhone

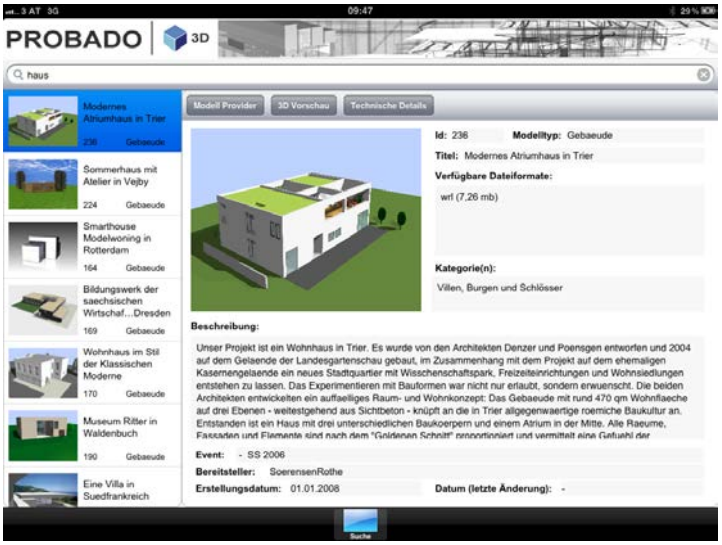


Abbildung 8.10: Detailansicht iPad

Abbildung 8.11: Detailansicht iPad

(a) 3D Darstellung iPad (b) 3D Darstellung Wireframe iPad

Abbildung 8.12: 3D Darstellung iPad Wireframe und Standard

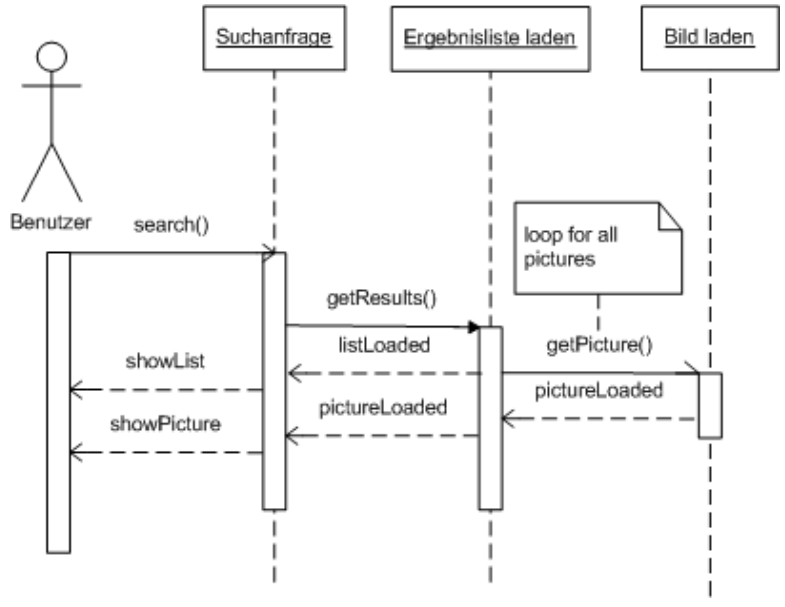


Abbildung 8.13: Suchabfrage iOS

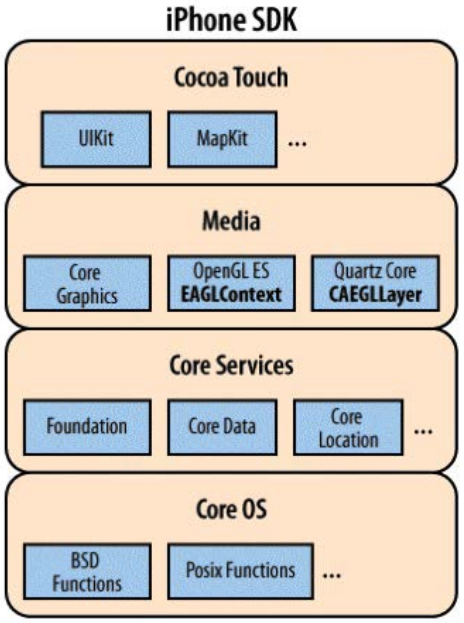


Abbildung 8.14: iPhone/iPad SDK [Rid10a]

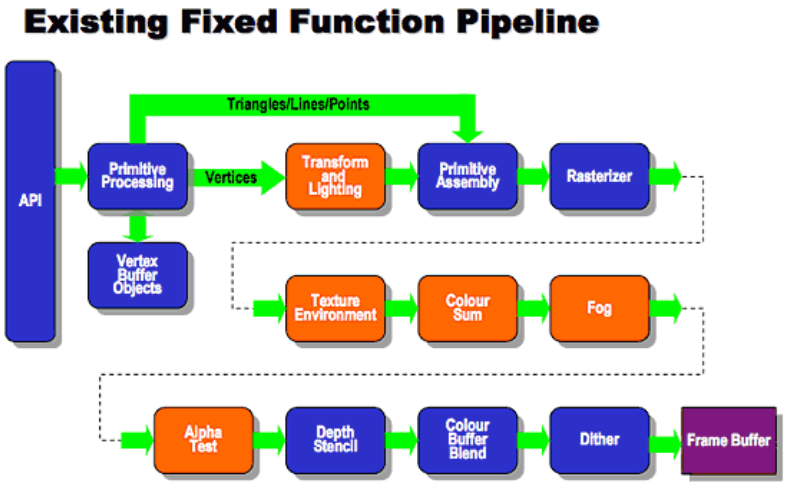


Abbildung 8.15: OpenGL ES 1.1 Pipeline [Ope12]

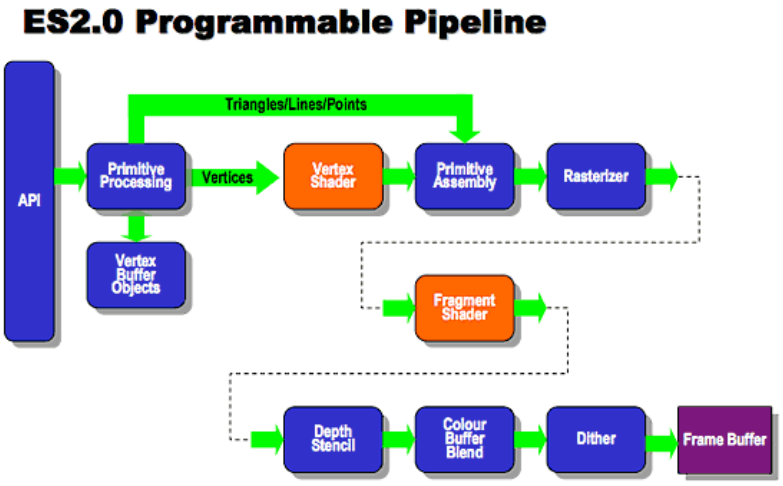


Abbildung 8.16: OpenGL ES 2.0 Pipeline [Ope12]

Kapitel 9

Lessons Learned

"Erfahrung nennt man die Summe all unserer Irrtümer" - ein Zitat von Thomas Edison, das auch in Bezug auf die vorliegende Arbeit von Bedeutung ist. Eine im Rahmen von Softwarentwicklungsprojekten häufig gemachte Erkenntnis ist, dass man erst nach Abschluss des Projektes weiß, welches Design und welche Entwicklungsumgebung man hätte wählen sollen, wenn man denn nur bei Projektbeginn bereits denselben Kenntnisstand gehabt hätte, wie nach Beendigung des Projektes. *"Was würde ich ändern, könnte ich das Projekt mit meinem aktuellen Wissen neu entwickeln? Was bleibt langfristig von der geleisteten Arbeit erhalten? Was habe ich für zukünfige Projekte dazugelernt?"* lauten die Kernfragen, die sich auch im Rahmen dieser Arbeit stellen.

Was habe ich dazugelernt?

- *"Softwarentwicklung ist ein zyklischer Prozess"*. Ein Satz, der immer wieder im Rahmen von Lehrveranstaltungen zum Thema Grundlagen der Informatik vorkommt, wenn der Student mit der Evolution der Software Prozessmodelle, beginnend beim klassischen Wasserfallmodell bis hin zu agilen Methoden der Softwarentwicklung wie Scrum oder Extreme Programming, konfrontiert wird. Auch im Rahmen dieses Projektes hat es sich gezeigt, dass es ein Irrtum ist, zu glauben, dass der perfekte Prototyp am Ende einer linearen Entwicklungsphase steht. Basierend auf dem Ursprungsdesign wurde in der zweiten Projektpahse von Monika Alter mit dem Cocoon-Framework ein erster CORE Protoyp umgesetzt, der dann im Rahmen dieser Arbeit nach Anpassung des Designs in C# reimplementiert wurde und sicher in Zukunft - da es sich um ein Produktivsystem handelt - noch weiteren Phasen der Erweiterung und Reimplementierung unterworfen sein wird. Wie die Recherche zum theoretischen Teil dieser Arbeit gezeigt hat, ist der Lebenszyklus einer Software untrennbar mit ständiger Weiterentwicklung und Verbesserung verbunden. Von den untersuchten verwandten Projekten sind nur diejenigen noch aktiv, die auch produktiv eingesetzt werden, im Gegensatz zu den reinen Forschungsprototypen, die zwar neue Erkenntnisse bieten, nach Beendigung des jeweiligen Projektes aber langsam in der Versenkung verschwinden.

- *"Je besser das Design, desto einfacher die Implementierung"*. Das in den ersten beiden Projektphasen erarbeitete Design war klar spezifiziert und erforderte nur geringe Anpassungen. Darauf aufbauend und auf den Erfahrungen, die bei der Implementierung des ersten Prototypen gemacht wurden, konnte die Reimplementierung problemlos und zügig abgeschlossen werden.

- *"Die eingesetzten Tools und Entwicklungsumgebungen müssen in Relation zur Teamstruktur stehen"*. Je größer die Mitarbeiterfluktuation, desto stabiler und verbreiteter muss das verwendete Framework sein, da sonst die Produktivität zu sehr leidet. Ein wesentliches Problem bei der ersten Implementierung mit Apache Cocoon waren die Komplexität und die langen Einarbeitungszeiten in das Cocoon Framework, sowie die häufigen Releasewechsel und Versionsinkompatibilitäten. Da PROBADO-Core als universitäres Projekt zu weiten Teilen nicht von fix angestellten Mitarbeitern, sondern von Studenten im Rahmen von Seminar- bzw. Masterarbeiten umgesetzt wurde, kam es während der Projektlaufzeit zu einem beständigen Mitarbeiterwechsel. Jeder neue Mitarbeiter musste sich einerseits ins Cocoon Framework einarbeiten, andererseits auf bestehenden Codebestandteilen aufbauen und diese nach jedem Releasewechsel an das geänderte Framework anpassen. Da es weder einen einheitlichen Codingstandard, noch klare Richtlinien gab, führte

das in relativ kurzer Zeit zu unübersichtlichem, kaum wartbarem Code. Dies änderte sich erst nach dem Umschwenken auf kommerzielle Entwicklungsumgebungen, deren weite Verbreitung die Einarbeitungszeit radikal reduzierte und die auch die nötige Stabilität aufwies, um ohne Expertenwissen produktiv arbeiten zu können.

- *"Kommunikation zwischen den einzelnen Projektmitarbeitern ist unerläßlich"*. Vor allem wenn ein Projekt auf verschiedene Standorte und unterschiedliche Projektgruppen verteilt ist, ist es extem wichtig, dass die Mitarbeiter regelmäßig miteinander in Kontakt stehen, wichtige Informationen weitergegeben werden und vor allem bei Codeteilen, die von unterschiedlichen Gruppen bearbeitet werden, jederzeit geklärt ist, wer wann warum und mit welcher Technologie an welchem Codeteil arbeitet. Als Negativbeispiel wäre hier zu nennen, dass während der zweiten Projektphase ein erster Prototyp der Administrationskonsole unter Einsatz des Cocoon Frameworks entstand, während parallel Monika Alter mit dem selben Framework an ihrer Diplomarbeit arbeitete, zwischen uns aber kein Kontakt bestand. Ein Austausch von Wissen über das eingesetzte Framework und eine Abstimmung in bestimmten Bereichen wäre rückblickend sicher von Vorteil gewesen.
- *"Die Bedürfnisse und Ziele der Endanwender müssen bereits zu Projektbeginn abgeklärt werden"*. In der heutigen Zeit steht ein System/Programm in den seltensten Fällen für sich alleine. Jede zusätzliche Applikation, die eine Organisation einsetzt, muss in die bestehende EDV-Lanschaft eingepasst werden. Es muss auf die Fähigkeiten und Möglichkeiten der Mitarbeiter des Kunden/Endanwenders Rücksicht genommen werden, sodass sich der Wartungsaufwand für diesen im Rahmen hält. Es macht z.B. keinen Sinn, aus Kostengründen ein System basierend auf Open-Source Software einzusetzen, das zwar Lizenzgebühren spart, aber zusätzlichen Wartungsaufwand verursacht, der teuer extern zugekauft werden muss. Daher ist es sinnvoll, bereits zu Projektbeginn folgende Fragen zu analysieren:
  - welche Technologien werden bereits eingesetzt?
  - sollen diese beibehalten werden oder besteht die Bereitschaft, unter Umständen sogar der Wunsch, gegebenenfalls einen Technologiewechsel durchzuführen.
  - welche Mindestanforderungen/Garantieansprüche werden an die Software gestellt (Stabilität/Wartung).
  - welche Möglichkeiten der Systembetreuung stehen zur Verfügung.

Was bleibt langfristig erhalten? Die wesentlichsten Bestandteile einer Software sind das grundlegende Design und die Schnittstellen nach außen. Bereits in den ersten Projektphasen wurde sehr viel Mühe in das Systemdesign sowie die Spezifikation der WSDL-Interfaces für den Datenaustausch und die Kommunikation zwischen den Spezialrepositorien verwendet. Daher wurden diese Bestandteile von PROBADO im Zuge der Reimplementierung nur geringfügig angepasst und blieben im Wesentlichen erhalten.

Was würde ich ändern? Zur ersten Frage ist zu sagen, dass der praktische Teil der vorliegenden Arbeit in der finalen Projektphase von PROBADO angesiedelt war, zu einem Zeitpunkt in dem die meisten grundlegenden Designentscheidungen bereits getroffen waren. Wenn überhaupt, sind Änderungen in dieser Phase nur mehr verbunden mit sehr viel Aufwand und hohen Kosten möglich. Aus diesem Grund müssten sämtliche Änderungen bereits in früheren Phasen durchgeführt werden. Zu erwähnende Punkte wären hier z.B.

- *Definition der Schnittstellen und Abstimmung der Fertigstellungstermine:*
  Um die Schnittstellen zwischen den einzelnen Bereichen sowie die Webservices intensiv testen zu können, wäre es erforderlich gewesen, die Fertigstellungstermine für einzelne Komponenten teamübergreifend abzustimmen und gemeinsame Testtermine zu vereinbaren und durchzuführen. Eine Import- oder Anfrageschnittstelle ist schwer zu testen, wenn das Gegenstück zum Zeitpunkt der Fertigstellung noch nicht zur Verfügung steht und simuliert werden muss.
- *Einheitliches Corporate Design:*
  Die Suchmasken für die Detailsuche in den Bereichen 3D und Musik sind in bezug auf Design und Bedienung sehr unterschiedlich gestaltet. Hier wäre bereits die Einführung eines gemeinsamen, übergreifenden Designs von Vorteil.

Kapitel 10

Zusammenfassung

Diese Masterarbeit befasst sich mit den theoretischen Grundlagen des Designs und der Implementierung digitaler Bibliotheken, sowie im speziellen mit dem Aufbau einer Middleware (CORE) Schicht für den Einsatz im PROBADO System, eines Bibliothekssystems für nicht textuelle Dokumente. Eine weitere Aufgabe bestand in der Umsetzung einer App für den Zugriff auf PROBADO über mobile Endgeräte (iPad/iPhone).

Ziel des PROBADO Projektes ist es, den gesamten bibliothekarischen Arbeitsprozess, beginnend beim Erwerb der Objekte, über die Erschließung bis hin zur Speicherung und zum Abruf, abzudecken. Das Projekt teilt sich in zwei Teilbereiche, Musik und 3D mit dem CORE Bereich als Verbindungsglied, wobei die hier vorliegende Arbeit zu gleichen Teilen im CORE und im 3D Bereich angesiedelt ist. Sie ist Teil der letzten Projektphase und baut im CORE Bereich auf der bereits im Zuge der Diplomarbeit von Monika Alter geleisteten Vorarbeit auf. Die Hauptziele der praktischen Arbeit im CORE Bereich bestanden darin, den vorab beschlossenen Technologiewechsel durchzuführen, sowie einen funktionsfähigen Prototypen zu schaffen. Da zum Zeitpunkt des Beginns dieser Arbeit bereits die meisten Designentscheidungen getroffen waren und auf bestehende Codeteile Rücksicht genommen werden musste, war der Gestaltungsspielraum bei der praktischen Umsetzung in vielen Bereichen stark eingeschränkt. Für PROBADO 3D waren einerseits Anpassungen im Datenbankbereich durchzuführen, andererseits Applikationen für den Zugriff auf das Webservice über mobile Endgeräte zu schaffen.

Um nicht nur den im Rahmen der praktischen Arbeit betroffenen Teilbereich, sondern auch das Umfeld, in dem die Arbeit angesiedelt ist, zu beleuchten, wird in den ersten Kapiteln näher auf grundlegende Begriffsdefinitionen sowie die theoretischen Grundlagen digitaler Bibliotheken eingegangen. Es wird versucht, das Konzept einer digitalen Bibliothek, deren Eigenschaften und Inhalte zu definieren. Weiters werden unterschiedliche Architekturformen und technische Umsetzungsmöglichkeiten untersucht, die anhand von Beispielen existierender digitaler Bibliothekssysteme näher erläutert werden. Im Anschluss an diese einführenden Kapitel folgt ein Vergleich ausgewählter digitaler Bibliothekssysteme mit PROBADO. Die eben erwähnten Kapitel bilden sozusagen den theoretischen Rahmen in dem die nun folgende detaillierte Beschreibung von PROBADO, insbesondere des CORE Bereiches, eingebettet ist. Da die Systemarchitektur zu Beginn dieser Arbeit bereits feststand, wird hier hauptsächlich auf die geänderte Technologie, sowie die Anpassungen, die im Zuge der Implementierung vorgenommen werden mussten, eingegangen. Aufbau und Funktionsweise des CORE-Prototypen werden detailliert beschrieben. Aufgrund von sich ergebenden Synergien erschien es als sinnvoll, Teile der für den CORE-Bereich entworfenen Architektur auch für den 3D Bereich zu übernehmen. Zusätzlich war auch hier ein Re-Design des Datenbankmodells sowie eine Adaption zur Unterstützung von Mehrsprachigkeit vonnöten. Daher wurden diese Aufgaben ebenfalls im Rahmen der praktischen Umsetzung durchgeführt. Die vorgenommenen Änderungen im CORE und 3D Bereich werden in den Kapiteln sechs und sieben beschrieben. Kapitel acht beschreibt Design und Umsetzung der mobilen Applikationen die für den Zugriff auf PROBADO 3D geschaffen wurden. Kapitel neun beschreibt die im Rahmen des Umsetzungsprozesses gewonnenen Erkenntnisse.

10.1 Schlussfolgerung

Wie bereits in der Zusammenfassung erwähnt, war die praktische Umsetzung in der finalen Projektphase von PROBADO angesiedelt, zu einem Zeitpunkt als sämtliche Designentscheidungen bereits ge-

troffen waren, und abgesehen von der Optimierung des Datenbankmodells, wenig Gestaltungsspielraum bestand. Der CORE-Prototyp (Frontend, Webservice, Datenbank, Schnittstellendefinitionen) wurde in Zusammenarbeit mit einem Mitarbeiter der TU-Darmstadt, sowie der Universität Bonn (Webrahmen für das Frontend) fertiggestellt und getestet. Die Umgestaltung des 3D-Datenbankmodells erfolgte in Zusammenarbeit mit Mitarbeiten des 3D Bereichs. Ca. 70% der erbrachten praktischen Arbeit bewegten sich im Bereich Datenbankdesign, Optimierung von SQL-Abfragen, sowie PLSQL-Code. Es gelang zeitgerecht, einen funktionsfähigen Prototypen fertigzustellen, der die notwendigen Grundfunktionalitäten aufweist. Teile der geplanten Funktionalitäten wie z.B. Benutzerfeedback, Abfrageprotokollierung, Personalisierung oder statistische Auswertungen von Suchbegriffen sind datenbankseitig vorbereitet, wurden aber applikationsseitig nicht umgesetzt um die Stabilität des Gesamtsystems zu diesem Zeitpunkt nicht zu gefährden. Die Benutzerverwaltung (Administrationskonsole) wurde fertiggestellt, wird aber zur Zeit ausschließlich im 3D Bereich eingesetzt. Die geplante Rechteverwaltung wurde nicht umgesetzt, da beschlossen wurde, dass PROBADO vorerst ausschließlich als Suchmaschine dienen soll, die tatsächliche Zugriffskontrolle auf die gefundenen Objekte, die ja häufig dem Urheberrecht unterliegen, von dem jeweiligen angeschlossenen externen Repository zu kontrollieren ist.

Die dem CORE-Bereich zugedachte Hauptaufgabe ist es, als Bindeglied zwischen den angeschlossenen Spezial-Repositorien zu dienen und bereichsübergreifende Abfragen zu ermöglichen. Da mit dem Wegfall des E-Learning Bereichs mit 3D und Musik nur mehr zwei Fachbereiche mit sehr unterschiedlichen Inhalten zur Verfügung standen, war mit Ausnahme von Demonstrationsbeispielen, kaum mehr Bedarf für Abfragen auf beide Bereiche vorhanden. Auch war in diesem Stadium nur eine geringe Anzahl externer Repositorien angeschlossen, sodass es dem Benutzer vernünftiger erschien, ein Spezial-Repository auszuwählen und die Abfragen direkt an dieses zu schicken. Ein weiterer großer Vorteil dieser Vorgangsweise war, dass die speziellen Abfrageinterfaces der Spezialbereiche, die ja die eigentliche Stärke von PROBADO darstellen, nur in diesem Kontext zur Verfügung stehen. Aktuell wird der CORE somit eher als Demonstrationssystem verwendet und im praktischen Einsatz meist umgangen. In Bezug auf Performance kann keine wirkliche Aussage gemacht werden, da für den Prototypen einerseits nur eine MS SQL-Server Express Edition (mit den bekannten Einschränkungen) verwendet wurde, andererseits nur eine beschränkte Menge an Testdaten zur Verfügung stand (mehrere tausend Datensätze aus dem 3D Bereich, sowie einige hundert Datensätze aus dem Musik-Bereich), die manuell über ein Skript importiert werden mussten, da zum Testzeitpunkt die entsprechenden Webservices für den Export der Daten aus den Spezialrepositorien nicht zur Verfügung standen. Es wurde zwar versucht, die Abfragen zu optimieren, und sie konnten auf dem Testsystem auch sehr performant durchgeführt werden, für einen tatsächlichen Produktiveinsatz müssten aber sicherlich noch Nachjustierungen getroffen werden.

Die Zusammenarbeit innerhalb des CORE-Teams, sowie mit dem 3D Team hat sehr gut funktioniert. Was sicher auch teilweise darin begründet war, dass zum Zeitpunkt des Einstiegs in das Projekt die Spezifikationsphase bereits abgeschlossen war und somit wenig Koordinationsbedarf bestand. Abschließend ist zu sagen, dass die Hauptziele des Projekts, die Umsetzung des Technologiewechsels, sowie die Fertigstellung eines funktionierenden CORE Prototypen erreicht wurden. Weitere Ziele, wie der Ausbau der Rechteverwaltung, der Einbau einer leistungsfähigen Session-Verwaltung oder der Produktiveinsatz wurden im Zuge der Implementierung vorerst auf Eis gelegt, da aufgrund der Unterschiedlichkeit der angeschlossenen Spezialbereiche kein wirklicher Bedarf dafür vorhanden war und somit beschlossen wurde, die vorhandenen Entwicklungsressourcen an anderer Stelle effektiver einzusetzen. Das ist einerseits schade, da somit der eigentliche Zweck des COREs als Bindeglied und Zentrum des Systems nicht erreicht wurde, andererseits verständlich, da die eigentliche Stärke des PROBADO Systems ja in den Spezialbereichen und hier insbesondere in den innovativen Methoden der automatischen Metadatenextraktion sowie in den speziellen Abfrageinterfaces liegt.

10.2 Ausblick

Ein CORE Prototyp mit den spezifizierten Grundfunktionalitäten wurde umgesetzt und getestet. Zusätzliche Bereiche, wie Session Handling, eine Benutzer- und Rechteverwaltung oder Feedbackmechanismen, wurden nicht oder nur teilweise umgesetzt. So wurde das Modul für die Benutzerverwaltung zwar implementiert, aber nicht eingebunden. Die Personalisierung von Suchabfragen wurde datenbankseitig vorbereitet, aber applikationsseitig nicht umgesetzt. Diese Erweiterungen könnten in einer späteren Version ergänzt werden, inklusive der dafür notwendigen Anpassungen an den CORE-Webservices sowie den Schnittstellen. Da für den Prototyp eine kostenlose Express Edition von MS SQL-Server verwendet wurde, für einen Produktiveinsatz jedoch eine kommerzielle Lizenz vonnöten wäre, könnte unter Um-

ständen ein Wechsel zu Postgres-SQL als DBMS in Betracht gezogen werden. Die dadurch erzielbare Kostenersparnis müsste in Relation zum Anpassungsaufwand für die CORE-Webservices gesehen werden, die durch den Einsatz von LINQ2SQL zur Zeit auf MS SQL-Server als DBMS beschränkt sind. Eine Anbindung von anderen DBMS wie Postgres oder MySQL ist durch die Einbindung von externen Bibliotheken wie z.B. *DBLinq*[1] zwar auf einfache Weise möglich, bietet aber nicht immer 100%-ige Unterstützung und kann in manchen Fällen Codeänderungen erforderlich machen. [Bel12]

Weitere, mögliche Erweiterungen, könnten in den Bereich der Mehrsprachigkeit fallen und zwar einerseits im Rahmen der Anpassung der Datenbankstruktur und der Suchmechanismen (Volltextsuche, Übersetzung von Inhalten) – hier könnte zum Beispiel das Apache UIMA-Projekt (Unstructured Information Management) interessante Ansätze liefern. Andererseits durch automatische Übersetzung von Suchbegriffen und Resultaten im Webinterface. [UIM10] Ebenfalls interessante Perspektiven für die bereichsübergreifende Suche könnten sich im Bereich des Einsatzes von Ontologien ergeben.

Wesentliche Ausbaupotentiale ergeben sich im Bereich der mobilen Applikationen. Hier könnte man einerseits die für Windows Phone 7 und iOS (iPad/iPhone) geschaffenen Apps um zusätzliche in der Browser-Version bereits vorhandene Funktionalitäten ergänzen, sowie andererseits das Portfolio um eine App für Android erweitern.

[1] *http://code2code.net/DB_Linq/*

Literaturverzeichnis

[ABC*06] AGOSTI M., BISCHOFS L., CANDELA L., CASTELLI D., FERRO N., HASSELBRING W., MOUMOUTZIS N., SCHULDT H., WEIKUM G., WURZ M., ZUZULA G.: D1.1.1 : Evaluation and Comparison of the Service Architecture, P2P, and Grid Approaches for DLs. 15, 21

[AH11] ALEM L., HUANG W.: *Recent Trends of Mobile Collaborative Augmented Reality Systems*. SpringerLink : Bücher. Springer, 2011. 77

[Alt09] ALTER M.: *Architektur eines offenen, verteilten Suchdienstes in nicht-textuellen Dokument-Repositorien*. Master's thesis, Technische Universität Graz, Institut für Computer Graphik und Wissensvisualisierung, Abschluss voraussichtlich: Oktober 2009. Begutachter/Betreuer: Dipl.-Inform. Dr.-Ing. Sven Havemann. 1, 3, 36, 41, 46, 47, 49, 50, 57, 66

[And13] ANDROID: Android What's new, URL: http://www.android.com/whatsnew/ Abgerufen am: 13.04.2013, 2013. 75

[App13] APPLE: GLKit Framework Reference http://developer.apple.com/library/mac/#documentation/GLkit/Reference/GLKit_Collection/_index.html, 2013. 79

[Aus12] AUSTRIA F.: PDF 3D, URL: http://www.fraunhofer.at/vc/forschung/pdf3d/ Abgerufen am: 18.01.2012, 2012. 76

[Bay02] BAYER T.: *REST Web Services*. Tech. rep., Orientation in Objects GmbH, 2002. 8

[BBC*] BINDING C., BRETTLECKER G., CATARCI T., CHRISTODOULAKIS S., CRECELIUS T., GIOLDASIS N., CHRISTIAN JETTER H., KACIMI M., MILANO D., RANALDI P., REITERER H., JÖRG SCHEK H., SCHULDT H., TUDHOPE D., WEIKUM G.: DelosDLMS: Infrastructure and Services for Future Digital Library Systems. 24, 25, 100

[BBC*10] BERNDT R., BLÜMEL I., CLAUSEN M., DAMM D., DIET J., FELLNER D. W., FREMEREY C., KLEIN R., KRAHL F., SCHERER M., SCHRECK T., SENS I., THOMAS V., WESSEL R.: The PROBADO Project - Approach and Lessons Learned in Building a Digital Library System for Heterogeneous Non-textual Documents. In *European Conference on Digital Libraries (ECDL)* (Sept. 2010), pp. 376–383. 1, 31, 33, 42, 100

[BBK*09] BERNDT R., BLÜMEL I., KROTTMAIER H., WESSEL R., SCHRECK T.: Demonstration of User Interfaces for Querying in 3D Architectural Content in PROBADO 3D. In *Research and Advanced Technology for Digital Libraries* (2009), pp. 491–492. 31, 32, 100

[BBWS09] BERNDT R., BLÜMEL I., WESSEL R., SCHRECK T.: Demonstration of User Interfaces for Querying in 3D Architectural Content in PROBADO3D. In *ECDL* (Sept. 2009), vol. 5714 of *Lecture Notes in Computer Science*, Springer. 29, 30, 31, 100

[Bel12] BELIN C.: Stackoverflow, URL: http://stackoverflow.com/questions/4855095/linq-to-sql-with-postgresql Abgerufen am: 10.04.2012, 2012. 92

[Ber12] BERNDT R.: Probado 3D Mobile Prototype, 2012. 80, 81, 82, 101

[BKHS09] BERNDT R., KROTTMAIER H., HAVEMANN S., SCHRECK T.: The PROBADO-Framework: Content-based Queries for Non-textual Documents. In *ELPUB 2009: 13th International Conference on Electronic Publishing* (2009), p. 17. 30

[Bom12] BOMFIN D.: NinevehGL, URL: http://nineveh.gl/ Abgerufen am: 18.01.2012, 2012. 78

[Bor00] BORGES J.: *Poema de los Dones*. Artesanías Gráficas, 2000. 1

[BRS*09] BRETTLECKER G., RANALDI P., SCHEK H.-J., SCHULDT H., SPRINGMANN M.: ISIS & OSIRIS: a Process-Based Digital Library Application on top of a Distributed Process Support Middleware. 24

[Buc12] BUCK E.: *Learning OpenGL ES for iOS: A Hands-on Guide to Modern 3D Graphics Programming*. Learning. Pearson Education, 2012. 80, 82

[BZ06] BENNING G., ZIMMER A.: SOA erfolgreich nutzen. *Javaspektrum*, 3 (2006), 22–26. 6, 7

[CCI*07] CANDELA L., CASTELLI D., IOANNIDIS Y., KOUTRIKA G., MEGHINI C., PAGANO G., ROSS S., SOERGEL D., AGOSTI M., DOBREVA M., KATIFORI V., SCHULDT H.: The DELOS Digital Library Reference Model. 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 100

[Cha08] CHARLES O.: *Einführung und Überblick: OpenGL: Vom Minimalbeispiel bis zur komplexen Applikation*. GRIN Verlag GmbH, 2008. 78

[Cla03] CLAUSEN J.: *REST in pieces*. Tech. rep., Universität Bielefeld, 2003. 8

[Cru10] CRUYSBERGHS S.: .NET - ADO.NET Entity Framework LINQ to Entities, URL: http://www.scip.be/index.php?Page=ArticlesNET12#EntityFramework Abgerufen am: 16.05.2010, 2010. 39, 40, 100

[DEL10] DELOS: DELOS Network of Excellence on Digital Libraries, URL: http://www.delos.info Abgerufen am: 10.05.2010, 2010. 1

[Deu11] DEUTSCHLAND S.: SEM Deutschland, URL: http://www.sem-deutschland.de/google-adwords-tipps/google-studie-zu-suchverhalten-in-deutschland/ Abgerufen am: 28.06.2011, 2011. 56

[Doe03] DOERR M.: The CIDOC CRM - An ontological approach to semantic interoperability of metadata. *AI Magazine 24* (2003), 2003. 4

[DUR10] DURASPACE: Fedora Commons http://www.fedora-commons.org, 2010. 27

[ea10] ET AL. L.: IT Wissen Online Lexikon der Informationstechnologie, URL: http://www.itwissen.info/definition/lexikon/service-oriented-architecture-SOA-SOA-Architektur.html Abgerufen am: 10.05.2010, 2010. 14

[ES12] ECKERT C., SCHNEIDER C.: *Smart Mobile Apps – Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse*. Xpert.press. Springer, 2012, ch. Smart Mobile Apps: Enabler oder Risiko?, pp. 193–207. 75, 76, 99

[Eur11] EUROPEANA: Europeana, URL: http://www.europeana.eu/portal/aboutus_background.html Abgerufen am: 28.06.2011, 2011. 28, 29, 100

[Fer08] FERRACCHIATI F. C.: *LINQ for Visual C# 2008*, 1 ed. Apress, Berkely, CA, USA, 2008. 38

[FK09] FISCHER M., KRAUSE J.: *ASP.NET 3.5*, 1 ed. Hanser, München, Deutschland, 2009. 37

[Foi09] FOIDL T.: *Erstellung einer Benutzer und Rollenverwaltung für PRODABO Core*. Tech. rep., Technische Universität Graz, Institut für Computer Graphik und Wissensvisualisierung, 2009. Betreuer: Harald Krottmaier. 47, 51, 52, 53, 54

[For10] FORUM O. A.: OAI for Beginners, URL: http://www.oaforum.org/tutorial/english/page1.htm Abgerufen am: 01.06.2010, 2010. 6

[Fra01] FRANCE R.: Effective, Efficient Retrieval in a Network of Digital Objects. 27

[GFF01] GONÇALVES M. A., FRANCE R. K., FOX E. A.: MARIAN: Flexible Interoperability for Federated Digital Libraries. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries, September 4-9, 2001, Darmstadt, Germany, Springer Lecture Notes in Computer Science 2163*. 2001, pp. 173–186. 26

[Gol12] GOLDSMITH K.: Kenneth Goldsmith Author's Page, URL: http://epc.buffalo.edu/authors/goldsmith/if_it_doesnt_exist.html Abgerufen am: 04.01.2012, 2012. 74

[Gro06] GROUP C. C. S. I.: *The Definition of the CIDOC Conceptual Reference Model*. Tech. rep., CIDOC CRM Special Interest Group, 2006. 5, 6, 100

[HKZZ99] HENRY S. M., KAFURA D. G., ZHAO J., ZHAO J.: Making Digital Libraries Flexible, Scalable and Reliable: Reengineering the MARIAN System in JAVA. In *Department of Computer Science, Blacksburg, VA, Master of Science Thesis* (1999), pp. 070499–204531. 27

[HM02] HAROLD E. R., MEANS W. S.: *XML in a nutshell*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002. 3

[IGD12] IGD F.: X3DOM, URL: http://www.igd.fraunhofer.de/Institut/Abteilungen/Visual-Computing-System-Technologies/Projekte/X3DOM Abgerufen am: 18.01.2012, 2012. 77

[IGN12] IGN: The History of Android, URL: http://www.ign.com/articles/2011/12/22/the-history-of-android Abgerufen am: 18.01.2012, 2012. 75

[Ini10] INITIATIVE D. C. M.: Dublin Core Metadata Initiative http://dublincore.org/documents/2006/12/18/dces, 2010. 4

[Ini11] INITIATIVE D. C. M.: Dublin Core Metadata Initiative http://dublincore.org/documents/2000/07/11/dcmes-qualifiers, 2011. 4

[KHP*06] KRUK S., HASLHOFER B., PIOTROWSK P., WESTERSKI A., WORONIECKI T.: The Role of Ontologies in Semantic Digital Libraries. In *The 5th European Networked Knowledge Organization Systems (NKOS) Workshop* (September 2006). 20

[KKSA07] KROTTMAIER H., KURTH F., STEENWEG T., APPELRATH H-J.AND FELLNER D. W.: PROBADO – A Generic Repository Integration Framework. In *Research and Advanced Technology for Digital Libraries, ECDL 2007*, vol. 4675. Springer, 2007, pp. 518–521. doi: 10.1007/978-3-540-74851-9. 29

[KM09] KRUK S. R., MCDANIEL B. (Eds.): *Semantic Digital Libraries*. Springer, 2009. 4, 5, 17, 21, 95, 97

[KR08] KRENZ M., RAMLOW M.: *Maschinelle Übersetzung und XML im Übersetzungsprozess: Prozesse der Translation und Lokalisierung im Wandel*. Frank & Timme, 2008. 3

[KWK09] KRUK S. R., WESTERSKI A., KRUK E.: Architecture of Semantic Digital Libraries. In Kruk and McDaniel [KM09], pp. 77–85. 21, 22, 23, 100

[Lag00] LAGOZE C.: Accommodating Simplicity and Complexity in Metadata: Lessons from the Dublin Core Experience, 2000. 4

[Lib10] LIBRARY O. D.: Oxford Digital Library, URL: http://www.odl.ox.ac.uk/metadata.htm Abgerufen am: 05.05.2010, 2010. 3, 4, 16

[M.F08] M.F.R.H.: *F+E-Vorhaben KEWA : Phase V, 2009/10*. KIT Scientific Publishing, 2008. 9

[Mic10] MICROSOFT: Microsoft Silverlight, URL: http://www.microsoft.com/silverlight/what-is-silverlight Abgerufen am: 18.05.2010, 2010. 37, 100

[Mic12a] MICROSOFT: DirectX, URL: http://msdn.microsoft.com/library/ee663275%28VS.85%29.aspx Abgerufen am: 13.02.2012, 2012. 78

[Mic12b] MICROSOFT: DirectX, URL: http://msdn.microsoft.com/library/ee663275%28VS.85%29.aspx Abgerufen am: 13.02.2012, 2012. 78

[Min04] MINOLI D.: *A Networking Approach to Grid Computing*. Wiley-Interscience, 2004. 15

[MR05] MEGHINI C., RISSE T.: BRICKS: A Digital Library Management for Cultural Heritage. *ERCIM News* (2005). 24, 25, 26, 100

[MSD09] MSDN: MSDN, URL: http://msdn.microsoft.com/de-de/library/bb386976.aspx Abgerufen am: 16.05.2010, 2009. 38, 39, 100

[Mue01] MUELLER J. P.: *Special Edition Using SOAP*. Que, 2001. 7

[oC11] OF CONGRESS L.: METS Metadata Encoding and Transmission Standard http://www.loc.gov/standards/mets, 2011. 5

[ODR10] ODRL: The ODRL Initiative, URL: http://odrl.net Abgerufen am: 03.05.2010, 2010. 27

[oI11] OF INFORMATION V.: VOI CC Standards & Normen http://www.voi.de/dokuwiki/doku.php/leitfaden:3_00_direkter_ecm_bezug:3_06_metadaten:3_06_08_mets, 2011. 5

[Ope11] OPENSEARCH: OpenSearch, URL: http://www.opensearch.org/Home Abgerufen am: 28.06.2011, 2011. 9, 28

[Ope12] OPENGL: OpenGL, URL: http://www.opengl.org/about/ Abgerufen am: 18.01.2012, 2012. 78, 82, 86, 87, 101

[Pro04] PROJECT B.: The Bricks Community, URL: http://www.brickscommunity.org Abgerufen am: 05.05.2010, 2004. 24

[pro11] PROBADO, URL: http://www.probado.de Abgerufen am: 01.02.2011, 2011. 77, 78, 101

[Qua12] QUAINE N.: SOAPUSER.COM, URL: http://www.soapuser.com/basics3.html Abgerufen am: 10.04.2012, 2012. 7

[RA08] RAINIE L., ANDERSON J.: The Future of the Internet III, URL: http://www.pewinternet.org/Reports/2008/The-Future-of-the-Internet-III.aspx Abgerufen am: 18.01.2012, 2008. 74

[Rid10a] RIDEOUT P.: *iPhone 3D Programming - Developing Graphical Applications with OpenGL ES*. O'Reilly, 2010. 81, 86, 101

[Rid10b] RIDEOUT P.: *IPhone 3D Programming: Developing Graphical Applications with OpenGL ES*. O'Reilly Series. O'Reilly Media, 2010. 78

[Ril10] RILEY J.: *A Glossary of Metadata Standards*. Tech. rep., Indiana University Libraries, 2010. 3

[SAF10] SCHERER M., ALTER M., FOIDL T.: *Core Documentation*. Tech. rep., CGV, 2010. 58, 61

[SB09] SENS I., BLÜMEL I.: Das PROBADO-Projekt. Integration von nicht-textuellen Dokumenten am Beispiel von 3D-Objekten in das Dienstleistungsangebot von Bibliotheken. *Zeitschrift fÂ¸r Bibliothekswesen und Bibliographie 56*, 2 (2009), 79–87. 31, 32, 100

[SCB07] SIMEONI F., CRESTANI F., BIERIG R.: The DILIGENT Framework for Distributed Information. *SIGIR 2007* (2007). 26

[Sch10a] SCHOOLS W.: W3C Schools SOAP Tutorial, URL: http://www.w3schools.com/soap/soap_syntax.asp Abgerufen am: 05.05.2010, 2010. 7, 8, 100

[Sch10b] SCHOOLS W.: W3C Schools W3C - RDF, URL: http://www.w3.org/RDF Abgerufen am: 05.05.2010, 2010. 6

[Sch10c] SCHOOLS W.: W3C Schools WSDL Tutorial, URL: http://www.w3schools.com/w3c/w3c_wsdl.asp Abgerufen am: 05.05.2010, 2010. 9

[Sch10d] SCHOOLS W.: W3C Schools WSDL Tutorial, URL: http://www.w3schools.com/w3c/w3c_wsdl.asp Abgerufen am: 10.05.2010, 2010. 2

[Sch12] SCHNELLER D.: Webservices mit iOS, URL: http://blog.codecentric.de/2012/01/soap-webservices-mit-ios/ Abgerufen am: 10.04.2012, 2012. 76

[SDK09] SYNAK M., DABROWSKI M., KRUK S. R.: Semantic Web and Ontologies. In Kruk and McDaniel [KM09], pp. 41–54. 19, 20

[SEL11] SELFHTML: Meta-Angaben zum Inhalt http://de.selfhtml.org/html/kopfdaten/meta.htm#dublin_core, 2011. 4

[Sem10] SEMATICWEB: Semanticweb, URL: http://semanticweb.org/wiki/Ontology Abgerufen am: 02.06.2010, 2010. 19

[Soe09] SOERGEL D.: Digital Libraries and Knowledge Organization. In Kruk and McDaniel [KM09], pp. 9–39. 18

[Som04] SOMMER D.: *Qualitätsinformationssysteme für E-Learning-Anwendungen*. Books on Demand, Norderstedt, 2004. 4

[SR07] SORENSEN J., ROUKOS S.: Rethinking Full-Text Search for Multilingual Databases. 73

[Sta10] STANFORD K.: KSL Stanford, URL: http://www.ksl.stanford.edu/projects/wine/explanation.html Abgerufen am: 01.06.2010, 2010. 19

[Tec10a] TECH V.: MARIAN Digital Library Information System, URL: http://www.dlib.vt.edu/products/marian.html Abgerufen am: 03.05.2010, 2010. 26

[Tec10b] TECH V.: NDLTD, URL: http://www.ndltd.org Abgerufen am: 04.06.2010, 2010. 27

[Tec10c] TECHTARGET: SearchWinDevelopment.com Definitions, URL: http://searchwindevelopment.techtarget.com/sDefinition/0,,sid8_gci1296647,00.html Abgerufen am: 16.05.2010, 2010. 38

[TEX*10] TSAI F. S., ETOH M., XIE X., LEE W.-C., YANG Q.: Introduction to Mobile Information Retrieval. *IEEE Intelligent Systems 25*, 1 (2010), 11–15. 74, 75, 101

[Til09] TILKOW S.: REST - der bessere Webservice. *Java Magazin* (2009). 8

[Trä00] TRÄGER B.: *Wissenschaft online. Elektronisches Publizieren in Bibliothek und Hochschule*. No. Sonderheft 80 in Zeitschrift für Bibliothekswesen und Bibliographie : Sonderhefte ; 80. Frankfurt a.M.: Klostermann (2000), 2000. Das Buch ist in der Bibliothek des Kurt-Schwitters-Forums der FH Hannover ausleihbar. Signatur: BID 7151/224:2. 4

[UIM10] UIMA A.: Apache UIMA, URL: http://uima.apache.org Abgerufen am: 16.05.2010, 2010. 92

[Ves04] VESTAVIK O.: REAP A System for Rights Management in Digital Libraries. *Proceedings of the First International ODRL Workshp* (2004). 27

[W3C12] W3CRDF: W3C RDF Primer, URL: http://www.w3.org/TR/rdf-primer/ Abgerufen am: 10.04.2012, 2012. 7, 100

[WB04] WORMUTH B., BECKER P.: Introduction to Formal Concept Analysis. *2nd International Conference of Formal Concept Analysis* (2004). 17, 20, 100

[Web10] WEBUCATOR: Silverlight Tutorial, URL: http://www.learn-silverlight-tutorial.com Abgerufen am: 18.05.2010, 2010. 38, 100

[Wen12] WENDERLICH R.: OpenGL ES 2.0 for iPhone Tutorial, URL: http://www.raywenderlich.com/3664/opengl-es-2-0-for-iphone-tutorial Abgerufen am: 18.01.2012, 2012. 82

[X3D12] X3DOM: X3DOM, URL: http://www.x3dom.org/?page_id=9 Abgerufen am: 18.01.2012, 2012. 77

[ZZZY11] ZHOU Z., ZHU R., ZHENG P., YANG B.: *Windows Phone 7 Programming for Android and IOS Developers*. John Wiley & Sons, 2011. 78

Tabellenverzeichnis

4.1 Dokumentenformate . . . 32
4.2 Systemarchitektur . . . 33
4.3 Protokolle . . . 34
4.4 Programmiersprachen . . . 34
4.5 Plattformen . . . 35
4.6 Projektdaten . . . 35
4.7 Legende: F..Forschungsprojekt | P..Produktivsystem . . . 35

6.1 Metadaten . . . 46
6.2 SEARCHQUERY . . . 53
6.3 SEARCHTERM . . . 54
6.4 SEARCHORDER . . . 55
6.5 QUERYRESULT . . . 55
6.6 Metadatensatz Repository Registrierung . . . 58
6.7 Metadatensatz Queryengine Registrierung . . . 59
6.8 Parameter Repository Metadatenexport . . . 63
6.9 Abfrageparameter Metadatensuche . . . 67
6.10 Abfrageparameter inhaltsbasierte Suche . . . 67
6.11 Datenbankfelder Metadatensuche . . . 68
6.12 Datenbankfelder Volltextsuche . . . 68

8.1 Überblick über die SOAP Unterstützung auf aktuellen mobilen Plattformen [ES12] . . . 76
8.2 Überblick über die 3D Unterstützung auf aktuellen mobilen Plattformen . . . 79

Abbildungsverzeichnis

2.1 Abbildung von zeitbezogenen Informationen durch CRM. Quelle: [Gro06] . . . 6
2.2 RDF Graph für Eric Miller. Quelle: [W3C12] . . . 7
2.3 SOAP Nachricht - vgl. [Sch10a] . . . 8

3.1 Framework in drei Schichten. Quelle: [CCI*07] . . . 12
3.2 Das digitale Bibliotheksuniversum: Hauptkonzepte. Quelle: [CCI*07] . . . 13
3.3 Zusammenhang zwischen den einzelnen Abstraktionsebenen. Quelle: [CCI*07] . . . 14
3.4 Beziehungen zwischen den Hauptkonzepten einer digitalen Bibliothek. Quelle: [CCI*07] 16
3.5 Rollen der Benutzer im 3-Schichtmodell. Quelle: [CCI*07] . . . 17
3.6 Interaktion von Benutzergruppen. Quelle: [CCI*07] . . . 18
3.7 DL Referenzmodell (Concept Map zur Darstellung der Hauptkonzepte = Ressourcen. Quelle: [CCI*07]) . . . 19
3.8 Definition eines Informationsobjektes. Quelle: [CCI*07] . . . 20
3.9 Objekte, Attribute und Beziehungen bilden ein Konzept. Quelle: [WB04] . . . 20
3.10 High Level Komponenten einer semantischen digitalen Bibliothek. Vgl. [KWK09] . . . 22
3.11 Schichtarchitektur einer semantischen digitalen Bibliothek. Vgl. [KWK09] . . . 23

4.1 Überblick über Delos DLMS. Quelle: [BBC*] . . . 25
4.2 Komponenten eines BRICKS Knoten. Quelle: [MR05] . . . 26
4.3 Europeana Such Interface. Quelle: [Eur11] . . . 28
4.4 Europeana Darstellung von Suchergebnissen. Quelle: [Eur11] . . . 29
4.5 Europeana Verfeinerung von Suchergebnissen. Quelle: [Eur11] . . . 29
4.6 3-Schichtarchitektur des PROBADO Systems. Quelle: [BBWS09] . . . 30
4.7 Suche über Raumverbindungsgraphen. Quelle: [SB09], [BBK*09] . . . 31
4.8 Anfrage aufgrund einer 3D-Skizze für inhaltsbasierte globale Suche. Quelle: [SB09], [BBK*09] . . . 32
4.9 2D Ergebnis Visualisierung. Quelle: [BBC*10] . . . 33

5.1 Architektur von Silverlight. Quelle: [Mic10] . . . 37
5.2 Silverlight Processing. Quelle: [Web10] . . . 38
5.3 Ablauf einer LINQ2SQL Abfrage. Quelle: [MSD09] . . . 39
5.4 Aufbau des Entity Frameworks. Quelle: [Cru10] . . . 40

6.1 Grober Überblick über die PROBADO Systemstruktur. Quelle: [BBC*10] . . . 42
6.2 Webinterface einfache CORE Suche mit Ergebnisliste . . . 43
6.3 Detaildarstellung eines Ergebniseintrags . . . 44
6.4 Webinterface erweiterte CORE Metadatensuche . . . 44
6.5 Kommunikationsvorgänge zwischen CORE und Spezialrepositorien . . . 45
6.6 Listenmodell . . . 48
6.7 Repository Datenbankmodell . . . 49
6.8 Metadatenmodell . . . 51
6.9 Administratives Datenbankmodel. Quelle: [Foi09] . . . 52
6.10 Oberfläche der Administrationskonsole. Quelle: [Foi09] . . . 53
6.11 Struktureller Aufbau der Benutzer und Rechteverwaltung. Quelle: [Foi09] . . . 54
6.12 Struktur zur Protokollierung von Suchanfragen . . . 55
6.13 Zeitlicher Ablauf bei der Registrierung eines neuen Repository . . . 59

6.14 Zeitlicher Ablauf beim Austausch der Metadaten 61
6.15 Eingabe der Suchanfrage und der gewünschten Ergebnisreihung ins Suchinterface 64
6.16 Abfrage im CORE 66
6.17 Darstellung der Ergebnisliste im Frontend 66

7.1 Überarbeites 3D Datenbankmodell 71
7.2 Tabellenstruktur zur Ablage von Schlagwörtern 72

8.1 Teilgebiete der mobilen Informationsgewinnung [TEX*10] 75
8.2 PROBADO 3D Detailansicht [pro11] 77
8.3 Anzeigearten PDF 3D [pro11] 78
8.4 Probado 3D App für WP7 [Ber12] 80
8.5 Ergebnisliste und Detailansicht Probado 3D WP7 [Ber12] 81
8.6 3D VorschauProbado 3D WP7 [Ber12] 82
8.7 iProbado 3d 82
8.8 Ergebnisliste und Detailansicht iPad 83
8.9 Ergebnisliste, Detailansicht und 3D Darstellung iPhone 83
8.10 Detailansicht iPad 84
8.11 Detailansicht iPad 84
8.12 3D Darstellung iPad Wireframe und Standard 85
8.13 Suchabfrage iOS 85
8.14 iPhone/iPad SDK [Rid10a] 86
8.15 OpenGL ES 1.1 Pipeline [Ope12] 86
8.16 OpenGL ES 2.0 Pipeline [Ope12] 87