AWI
Alfred-Wegener-Institut
für Polar- und Meeresforschung

Hochschule Bremerhaven
University of Applied Sciences

— Diplomarbeit —
zur Erlangung des Grades
Diplom-Wirtschaftsinformatiker (FH)

Prototypenentwicklung und ökonomischer Leistungsvergleich von OAI- und Web Service-orientierten Architekturen zur Realisierung von Repositories für Publikationen

in einer wissenschaftlichen Großforschungseinrichtung

Bastian Onken
Matrikel-Nr.: 22213

30. September 2005

Betreut durch Prof. Dr. Dieter Viefhues–Veensma (Referent)
und Dr. Ana Macario (Koreferentin)

Vorgelegt von: **Bastian Onken**
Matr.-Nr.: 22213
Burgstraße 24a
27612 Stotel
+49 (160) 9864 2410
bonken@awi-bremerhaven.de

Referent: **Prof. Dr. Dieter Viefhues–Veensma**
Hochschule Bremerhaven
An der Karlstadt 8
27568 Bremerhaven
+49 (471) 4823 – 448
viefhues@uni-bremen.de

Koreferentin: **Dr. Ana Macario**
AWI
Stiftung Alfred-Wegener-Institut
für Polar- und Meeresforschung
Am Handelshafen 12
27570 Bremerhaven
+49 (471) 4831 – 1435
amacario@awi-bremerhaven.de

Version vom 30. September 2005, 08:25 Uhr
Dokument gesetzt in LaTeX 2ε
This is pdfTeX, Version 3.141592-1.20b (MiKTeX 2.4.1986)

Danksagung

Meinen Betreuern Frau Dr. Ana Macario und Herrn Professor Dr. Dieter Viefhues-Veensma möchte ich für die freundliche Überlassung des Themas und für ihren vielseitigen fachlichen Rat danken. Ich verdanke ihnen jede erdenkliche Fürsprache, hilfreiche Unterstützung und anregende Gespräche. Frau Dr. Macario möchte ich meinen besonderen Dank aussprechen. Sie hat mich in jeder Phase der Arbeit sehr sachkundig und richtungsweisend begleitet, mich stets ermuntert und viel Geduld gezeigt.

Nicht unerwähnt möchte ich jene lassen, die mir unermüdlich und immer freundlich im Rechenzentrum tatkräftige Hilfen waren, sowie alle, die mir zur praktischen Umsetzung und Fertigstellung der Arbeit verhalfen. Ihnen allen gilt mein herzlicher Dank.

Darüberhinaus möchte ich mich bei meiner Familie und besonders bei meinen Eltern bedanken, ohne die diese Diplomarbeit und die damit verbundene Studienlaufbahn kaum möglich gewesen wäre.

Am Meisten möchte ich allerdings meiner Freundin Jessica danken. Sie stand mir immer bestärkend beiseite und motivierte mich allezeit. Ohne sie währe das Ergebnis dieser Arbeit mit Sicherheit ein Anderes.

Kurzfassung

Die Forschungsinstitution Alfred-Wegener besitzt derzeit ein umfangreiches elektronisches Archiv für wissenschaftliche Publikationen. Diese sollen zukünftig im Sinne eines „Open Archives“ verfügbar gemacht werden. In dieser Arbeit wird ein Ansatz entwickelt, der Alternativen und Lösungen zu den gegenwärtig teuren und unflexiblen Archivierungsmethoden von Publikationen beschreibt. Dazu werden zwei Architekturen zur Realisierung von Open Access (*OAI-PMH* und *Web Services via SOAP*) technologisch, wie auch ökonomisch gegenübergestellt.

Nach dem ausführlichen Vergleich beider Architekturen schließt sich die Implementierung eines Prototyps an, der auf einer geeigneten Technologie zur Erlangung der Anforderungen von Open Access basiert und zur Lösung der Probleme des bestehenden Systems beiträgt. Im Abschluss werden zusätzlich zukünftig mögliche Entwicklungen diskutiert.

Abstract

This work deals with Open Access and its implementation at the Alfred Wegener Institute; Bremerhaven. The existing system for archiving scientific publications has proved to be expensive and to lack the flexibility required in the Open Archives-framework. Two alternative architectures compliant with Open Access concepts are presented: *OAI-PMH* and *Web Services via SOAP*.

A detailed comparison of both architectures is followed by the implementation of a prototype. This is based on a suitable technology for fullfilling the requirements of Open Access and contributes to the solution of the problems of the existing system. In the conclusion future developments are discussed.

Inhaltsverzeichnis

Abbildungsverzeichnis xiii

Tabellenverzeichnis xv

Quelltextverzeichnis xvii

1 Einleitung 1
- 1.1 Motivation . . . 1
- 1.2 Das Alfred-Wegener-Institut . . . 1
- 1.3 Ziel der Arbeit . . . 3
- 1.4 Aufbau der Arbeit . . . 4
- 1.5 Zeitlicher Rahmen . . . 4

2 Grundlagen 7
- 2.1 Organisation von Informationen . . . 7
  - 2.1.1 Metadaten . . . 7
  - 2.1.2 Dublin Core Metadata Element Set . . . 8
- 2.2 Open Access . . . 9
  - 2.2.1 Ideologie . . . 9
  - 2.2.2 Ziele . . . 10
  - 2.2.3 Open Archives . . . 10
  - 2.2.4 OAI Protocol for Metadata Harvesting (OAI-PMH) . . . 11
- 2.3 Web Services . . . 11
  - 2.3.1 Ziel . . . 11
  - 2.3.2 Ursprung . . . 12
  - 2.3.3 Technologie . . . 12
  - 2.3.4 Kombination der Technologien . . . 18
- 2.4 Repositories . . . 20
  - 2.4.1 Definition . . . 20
  - 2.4.2 Besondere Eigenschaften von Repositories . . . 20

2.4.3 Anforderungen . . . 21
2.4.4 Das Repository-System „Fedora“ . . . 21
2.4.5 Institutionelle Repositories . . . 23
2.5 Entwicklungsumgebung . . . 24
2.5.1 Apache HTTP Server . . . 24
2.5.2 Extensible HyperText Markup Language (XHTML) . . . 25
2.5.3 Cascading Style Sheets (CSS) . . . 25
2.5.4 Hypertext Preprocessor (PHP) . . . 26

3 Überlegungen zur Leistungsoptimierung 29
3.1 Konzepte zur Abfrage von Daten . . . 29
3.1.1 OAI-PMH . . . 30
3.1.2 Web Services . . . 37
3.2 Vergleich . . . 43
3.2.1 Gegenüberstellung der Konzepte . . . 43
3.2.2 Ergebnisse der Gegenüberstellung beider Konzepte . . . 47
3.2.3 Auswahl . . . 50

4 Anforderungen an das System 51
4.1 Zielbestimmung . . . 51
4.1.1 Muss-Kriterien . . . 51
4.1.2 Kann-Kriterien . . . 52
4.1.3 Abgrenzungskriterien . . . 52
4.2 Einsatz . . . 52
4.2.1 Anwendungsbereiche . . . 52
4.2.2 Zielgruppen . . . 52
4.2.3 Betriebsbedingungen . . . 53
4.3 Umgebung . . . 53
4.3.1 Software . . . 53
4.3.2 Hardware . . . 53
4.3.3 Orgware . . . 54
4.4 Funktionalität . . . 54
4.5 Daten . . . 54
4.6 Leistungen . . . 55
4.7 Benutzeroberfläche . . . 55
4.8 Qualitätsziele . . . 56
4.8.1 Geschwindigkeit . . . 56
4.8.2 Skalierbarkeit . . . 56
4.8.3 Wartbarkeit / Änderbarkeit . . . 56

5 Entwicklung eines Prototypen 57
5.1 Grundkonzeption . . . 57
5.1.1 Nutzungsszenarien . . . 57
5.1.2 Beschreibung des zukünftigen Nutzers . . . 58

5.1.3 Grundsätzliche Überlegungen zu Ein- und Ausgabemedien . 60
5.1.4 Zusammenstellung der Funktionen 62
5.1.5 Überlegungen zur Eingabe-/Ausgabeschnittstelle 67
5.2 Benutzeroberfläche 69
5.2.1 Farbe und visuelle Kontraste 69
5.2.2 Schrift 70
5.2.3 Grundlegendes Seitenkonzept 71
5.3 Pflege und Weiterentwicklung 72
5.4 Verbreitung 72

6 Implementierung 73
6.1 Server-Anwendung 73
6.1.1 Fedora Repository 73
6.1.2 Apache Webserver 83
6.1.3 Strategie zur Datenübernahme 86
6.1.4 Schnittstellen der Server-Anwendung 88
6.1.5 Funktionstest 89
6.2 Client Anwendung 95
6.2.1 Implementierung der Funktionalitäten 95
6.2.2 Konfiguration 96
6.2.3 Verbindung zur Datenhaltung 97
6.2.4 Integration der Client-Anwendung in die Systemarchitektur . 98
6.2.5 Realisierung gleichzeitiger Suchanfragen 100
6.2.6 Grenzen des Prototyps 104

7 Dokumentation der Anwendung 105
7.1 Elemente der Benutzeroberfläche 105
7.1.1 Banner 105
7.1.2 Navigation 105
7.1.3 Suchformular 106
7.1.4 Ergebnisanzeige 107
7.1.5 Validierung der Benutzeroberfläche 108
7.2 Beispielhafte Sitzung 109
7.2.1 Startseite 109
7.2.2 Suche nach Publikationen 110
7.2.3 Fehlerbehandlung 114
7.2.4 Informationen und Hilfe 116

8 Zusammenfassung und Ausblick 119
8.1 Repositories an wissenschaftlichen Einrichtungen 119
8.2 Ausblick 120
8.2.1 Web Service-basierte Repositories 120
8.2.2 SOAP-Client 121

9 Persönliches Resümee **123**

Abkürzungsverzeichnis **125**

Literaturverzeichnis **129**

Anhang 137

A Quelltext: Konfiguration Fedora-Repository **137**
A.1 fedora.fcfg . . . 137

B Quelltext: Repository-Datenimport **147**
B.1 LDAP2Fedora-Cronjob . . . 147
B.1.1 ePIC2fedora.php . . . 147
B.1.2 foxml_open.xml . . . 155
B.1.3 foxml_close.xml . . . 156

C Quelltext: SOAP-Client **157**
C.1 Interfaces . . . 157
C.1.1 interface.webService.php . . . 157
C.2 Objektklassen . . . 158
C.2.1 class.fedoraWebService.php . . . 158
C.2.2 class.pangaeaWebService.php . . . 162
C.2.3 class.condition.php . . . 166
C.3 Wrapper . . . 167
C.3.1 webServiceCaller.php . . . 167
C.4 Funktionsbausteine . . . 169
C.4.1 functions-search.inc.php . . . 169
C.4.2 functions-common.inc.php . . . 171
C.5 Einstellungen . . . 172
C.5.1 config.inc.php . . . 172
C.6 GUI . . . 174
C.6.1 header.inc.php . . . 174
C.6.2 footer.inc.php . . . 174
C.6.3 xhtmlheader.inc.html . . . 175
C.6.4 xhtmlbuilders.inc.php . . . 176
C.6.5 searchform.inc.php . . . 181
C.6.6 global.css . . . 182
C.6.7 index.php . . . 186
C.6.8 help.php . . . 187
C.6.9 links.php . . . 189
C.6.10 about.php . . . 190

Abbildungsverzeichnis

1.1 Organigramm des Alfred-Wegener-Instituts (vgl. [AWI05b]) 2
1.2 Projektgruppen des Alfred-Wegener-Instituts (vgl. [AWI05b]) 3
1.3 Projektplan 5

2.1 SOAP Envelope 15
2.2 Schema eines WSDL-Dokuments 16
2.3 Verknüpfungen der Typen, vgl. [KL04] 17
2.4 Web Services (vgl. [Kre04]) 19
2.5 Vereinfachte Architektur des Fedora Repository (vgl. [Fed05, Section 4]) 22
2.6 Interaktionsmöglichkeiten mittels Web Services (vgl. [Fed05, Section 4]) 23
2.7 PHP: Funktionsweise (vgl. [Wik05b]) 26

3.1 Grundlegende Funktionsweise von OAI-PMH (vgl. [Car03]) 31
3.2 Harvesting-Prozesse mehrerer Service- und Data-Provider (vgl. [Car03]) 31
3.3 Zwischenschalten eines Aggregatoren (vgl. [Car03]) 32
3.4 OAI-PMH: Überblick der Struktur und Funktionen (vgl. [Car03]) . 33
3.5 Grundlegende Funktionsweise der Kommunikation von Web Services 37
3.6 Funktionsweise von Abfragen bei Respositories mit SOAP-API . . . 38
3.7 Gegenüberstellung der Konzepte 44

5.1 Anwendungsfall: Publikation suchen 63
5.2 Anwendungsfall: Links anzeigen 64
5.3 Anwendungsfall: Hilfe anzeigen 65
5.4 Anwendungsfall: Informationen anzeigen 66
5.5 Grundlayout 71

6.1 Shell: fedora-start.sh 82
6.2 Startseite des integrierten Tomcat-Applikationsserver 83
6.3 Programmablaufplan Datenimport 87
6.4 Einfügen des Objektes durch die Administrationsoberfläche 90
6.5 Architektur des Systems 99
6.6 Programmablaufplan Publikationssuche 103

7.1 Banner 105
7.2 Navigation 106
7.3 Hover Effekt 106
7.4 Visited Link 106
7.5 Suchformular 106
7.6 Ergebnisse 107
7.7 Detailansicht 108
7.8 Startseite 109
7.9 Repository Auswahl 110
7.10 Anzeige der Ergebnisse 111
7.11 Weitere Ergebnisse anzeigen 112
7.12 Detailansicht eines Ergebnisses 113
7.13 Leere Eingabe 114
7.14 Keine Ergebnisse gefunden 115
7.15 Links 116
7.16 Verantwortliche 117
7.17 Hilfe 118

Tabellenverzeichnis

2.1 Dublin-Core Metadata Element Set 9

3.1 Fedora-API-A: findObjects 40
3.2 Fedora-API-A: resumeFindObjects 41
3.3 OAI-PMH versus Web Services: Anforderungen und Aufwand der Implementierung 44
3.4 OAI-PMH versus Web Services: Standardisierung 45
3.5 OAI-PMH versus Web Services: Suche und Art/Volumen der Datenübertragung 45
3.6 OAI-PMH versus Web Services: Implementierungskosten 46

5.1 Benutzergruppen 59
5.2 Festlegen der Schriftarten 71

6.1 PHP Parameter 84
6.2 Fedora OAI-API 88
6.3 Produktvergleich 104

Quelltextverzeichnis

2.1 Beispiel: XML-Dokument 13
2.2 Beispiel SOAP-Envelope 15
2.3 Beispiel WSDL 17
2.4 Programmbeispiel PHP 27
3.1 OAI-PMH Request 34
3.2 Beispiel OAI-PMH-Response 35
3.3 findObjects-SOAP-Request 41
3.4 findObjects-SOAP-Response 42
6.1 fedora.fcfg: Name des Repository 73
6.2 fedora.fcfg: Email-Adressen der Administratoren 74
6.3 fedora.fcfg: Speicherort für Objekte 74
6.4 fedora.fcfg: Temporäres Verzeichnis 74
6.5 fedora.fcfg: Speicherort für Datastreams 74
6.6 fedora.fcfg: Port 74
6.7 fedora.fcfg: Adresse des Servers 74
6.8 fedora.fcfg: Persistent Identifier Namespace 75
6.9 fedora.fcfg: Weitere Namespaces 75
6.10 fedora.fcfg: Character-Encoding 75
6.11 fedora.fcfg: Datenbank 76
6.12 fedora.fcfg: Verbindung zur Datenbank 76
6.13 fedora.fcfg: Datenbank-Treiber 76
6.14 fedora.fcfg: Minimale Verbindungen zur Datenbank 76
6.15 fedora.fcfg: Maximale Verbindungen zur Datenbank 76
6.16 fedora.fcfg: Hosts 77
6.17 fedora.fcfg: Maximale Anzahl der Ergebnisse 77
6.18 fedora.fcfg: Sessionlänge 77
6.19 fedora.fcfg: OAI-Data-Provider Name 77
6.20 fedora.fcfg: Domain 78
6.21 fedora.fcfg: Email-Adresse der Administration 78
6.22 fedora.fcfg: Friends 78
6.23 fedora.fcfg: Maximale Anzahl Ergebnisse 78
6.24 fedora.fcfg: Maximale Anzahl Kopfdaten 78
6.25 FedoraOAIProvider.java, Zeilen 136-138 79
6.26 FedoraOAIProvider-fixed.java, Zeilen 136-138 79

6.27 FedoraOAIProvider.java, Zeilen 380-404 80
6.28 FedoraOAIProvider-fixed.java, Zeilen 380-410 81
6.29 Setzen der Pfade 81
6.30 PHP Konfiguration 84
6.31 httpd.conf 85
6.32 FedoraOAIProvider.java, Zeile 79 85
6.33 FedoraOAIProvider-fixed.java, Zeile 79 85
6.34 Testobjekt Bra2005b 89
6.35 FOXML: awi:Bra2005b 91
6.36 Fedora SOAP-Request 92
6.37 Fedora SOAP-Response 93
6.38 class.fedoraWebService.php, Zeilen 91 - 98 98
6.39 Ergebnis einer Suchanfrage durch webServiceCaller.php – Suchwort „atlantic“ 100
A.1 fedora.fcfg 137
B.1 ePIC2fedora.php 147
B.2 foxml_open.xml 155
B.3 foxml_close.xml 156
C.1 interface.webService.php 157
C.2 class.fedoraWebService.php 158
C.3 class.pangaeaWebService.php 162
C.4 class.condition.php 166
C.5 webServiceCaller.php 167
C.6 functions-search.inc.php 169
C.7 functions-common.inc.php 171
C.8 config.inc.php 172
C.9 header.inc.php 174
C.10 footer.inc.php 174
C.11 xhtmlheader.inc.html 175
C.12 xhtmlbuilders.inc.php 176
C.13 searchform.inc.php 181
C.14 global.css 182
C.15 index.php 186
C.16 help.php 187
C.17 links.php 189
C.18 about.php 190

Kapitel 1

Einleitung

1.1 Motivation

Das Alfred-Wegener-Institut für Polar- und Meeresforschung in Bremerhaven ist eine Großforschungseinrichtung mit rund 780 Mitarbeitern weltweit. Die hier tätigen Wissenschaftlicher produzieren jährlich eine stetig anwachsende Menge von Publikationen aus allen Forschungsbereichen. Doch bei der Organisation dieser Veröffentlichungen stoßen die eingesetzten Systeme inzwischen an ihre Grenzen – die große Anzahl der Daten lässt sich immer schwieriger verwalten. Hinzu kommt, dass Richtlinien von „Open Access“[1] eingehalten werden müssen, welche laut der Berliner Erklärung[2] vorgeschrieben sind.

Um diese Probleme zu lösen wird seit einiger Zeit nach einer geeigneten Technologie zur professionellen Langzeitarchivierung von Publikationen jedweder Art gesucht. Es wurden bereits einige interessante Architekturen gefunden, welche die gestellten Anforderungen erfüllen könnten. Die Diplomarbeit setzt genau an dieser Stelle an: Ziel ist die Erarbeitung von Lösungsansätzen sowie die Entwicklung eines Prototyps, welcher den technischen und wissenschaftlichen Kriterien genügt und dabei auf zukunftsweisenden Technologien basiert.

1.2 Das Alfred-Wegener-Institut

Da diese Diplomarbeit im Rechenzentrum des Alfred-Wegener-Instituts für Polar- und Meeresforschung erarbeitet wurde, soll das folgende Profil einen kurzen Überblick über das Institut geben.

[1] Metakonzept, weltweit Publikationen frei zur Verfügung zu stellen; vgl. dazu Abschnitt 2.2
[2] s. Seite 10

Die Stiftung Alfred-Wegener-Institut führt wissenschaftliche Projekte in der Arktis, Antarktis und den gemäßigten Breiten durch. Sie koordiniert die Polarforschung in Deutschland und stellt die für Polarexpeditionen erforderliche Ausrüstung und Logistik zur Verfügung. Zu den Aufgaben in der Meeresforschung gehören die Nordseeforschung, Beiträge zum biologischen Monitoring in der hohen See, Untersuchungen zur Meeresverschmutzung und zu marinen Naturstoffen sowie meerestechnische Entwicklungen.

1980 wurde das Institut in Bremerhaven als Stiftung des öffentlichen Rechts gegründet. Die Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung umfasst das Alfred-Wegener-Institut für Polar- und Meeresforschung in Bremerhaven, die Forschungsstelle Potsdam (seit 1992), die Biologische Anstalt Helgoland (seit 1998) und die Wattenmeerstation Sylt. Sie ist Mitglied der Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren (HGF) und wird zu 90% vom Bundesministerium für Bildung und Forschung finanziert. Das Land Bremen ist mit 8% beteiligt, die Länder Brandenburg und Schleswig-Holstein mit je 1%. Die Stiftung hatte 2005 einen Etat von 100 Mio. Euro und beschäftigt rund 780 Mitarbeiterinnen und Mitarbeiter (vgl. [AWI05a]).

Folgendes Organigramm zeigt die interne Organisationsstruktur des Instituts:

Abbildung 1.1: Organigramm des Alfred-Wegener-Instituts (vgl. [AWI05b])

Neben den Fachbereichen existieren weitere Projektgruppen, die sich mit verschiedensten Themen beschäftigen. Zusammengefasst sind diese im MARCOPOLI-Forschungsprogramm:

Abbildung 1.2: Projektgruppen des Alfred-Wegener-Instituts (vgl. [AWI05b])

Obwohl sich die zentralen Themen der Einrichtung auf die Polar- und Meeresforschung konzentrieren, ist die Einbindung von Service-Abteilungen in einem Unternehmen dieser Größe unabdingbar. Institutseigene Dienstleistungsabteilungen bestehen neben Logistik und Forschungsplattformen. Zu nennen sind die Abteilungen Verwaltung, Allgemeinen Serviceeinrichtungen, Bibliothek, Technologietransfer und das Rechenzentrum.

Die Aufgabengebiete des Rechenzentrums, welches aus einem rund 30-köpfigen Team besteht, sind breit gefächert: Die Aufgabenbereiche erstrecken sich von grundlegendem IT-Support bis hin zu hoch performantem wissenschaftlichen Rechnen, beispielsweise Berechnungen zur Klimaforschung.

1.3 Ziel der Arbeit

Ziel der Arbeit ist die Entwicklung eines Prototyps, der die bestehende Datenhaltung für Publikationen (ePIC[3]) vollwertig ersetzen kann und eine Umgebung bietet, die den Anforderungen von „Open Access“ entspricht.

In diesem Rahmen soll untersucht werden, inwieweit sich die zur Verfügung stehenden Technologien eignen. Dieser Eignungstest soll in einer kurzen Studie, in der es um Fragen der Leistungsoptimierung geht, einfließen und ausgewertet werden. Zu Hilfe soll dazu das Repository-System „Fedora“ der New Yorker Cornell University kommen, welches Kern-Funktionen für das Erreichen der Aufgabe bereitstellt.

Am Ende dieser Arbeit ist ein lauffähiges Fedora-Repository-System eingerichtet worden. Des Weiteren ist die Entwicklung eines funktionsfähigen Prototyps einer

[3]ePIC: Electronic Publication Information Center

Client-Anwendung abgeschlossen, welcher Daten verschiedener Repositories aufbereitet darstellen kann.

1.4 Aufbau der Arbeit

Die vorliegende Arbeit beschreibt die Entwicklung des oben genannten Prototyps und dessen angeschlossene Bausteine zur Integration an die bestehenden Verhältnisse am Alfred-Wegener-Institut. Nach der Einleitung wird auf theoretische Grundlagen die das Thema betreffen eingegangen. Nach dieser grundlegenden Einführung werden Überlegungen zur Leistungsoptimierung angestellt, die schließlich zur Auswahl einer geeigneten Technologie für das zu erstellende Produkt führen.

Da für die Software-Entwicklung ein phasenorientiertes Vorgehensmodell gewählt wurde, beschreiben die darauf folgenden Kapitel die einzelnen Entwicklungsphasen. Aufbauend auf der Anforderungsanalyse mittels Pflichtenheft schließt sich die Modellierungsphase zur Entwicklung des Prototyps an. Darauf folgend wird auf die Implementierung der Arbeitsergebnisse eingegangen, welche mit der anschließenden Dokumentation der Anwendung endet.

Kapitel 8 beendet die Arbeit mit einer Zusammenfassung der Ergebnisse, welche schließlich in einem Ausblick mündet, in dem mögliche Perspektiven zur weiteren Verwendung angeführt werden.

1.5 Zeitlicher Rahmen

Die Arbeit wird im Zeitfenster des 8. Semesters des Studiengangs Wirtschaftsinformatik der Hochschule Bremerhaven angefertigt. Das Rechenzentrum des Alfred-Wegener-Instituts in Bremerhaven wird der Ort der Durchführung sein. Die Höhe der zu erbringenden Leistung ist vertraglich mit 80 Stunden pro Monat auf vier Monate festgelegt. Vorab sind grundlegende Vorbereitungen eingeplant, sowie eine Implementierung eines Repository-Systems. Begonnen wird mit der Diplomarbeit darauf folgend am 03. August 2005 – der späteste Termin zur Fertigstellung ist der 02. November 2005. Im Folgenden sind die einzelnen Arbeitspakete der Diplomarbeit auf einer Zeitachse dargestellt:

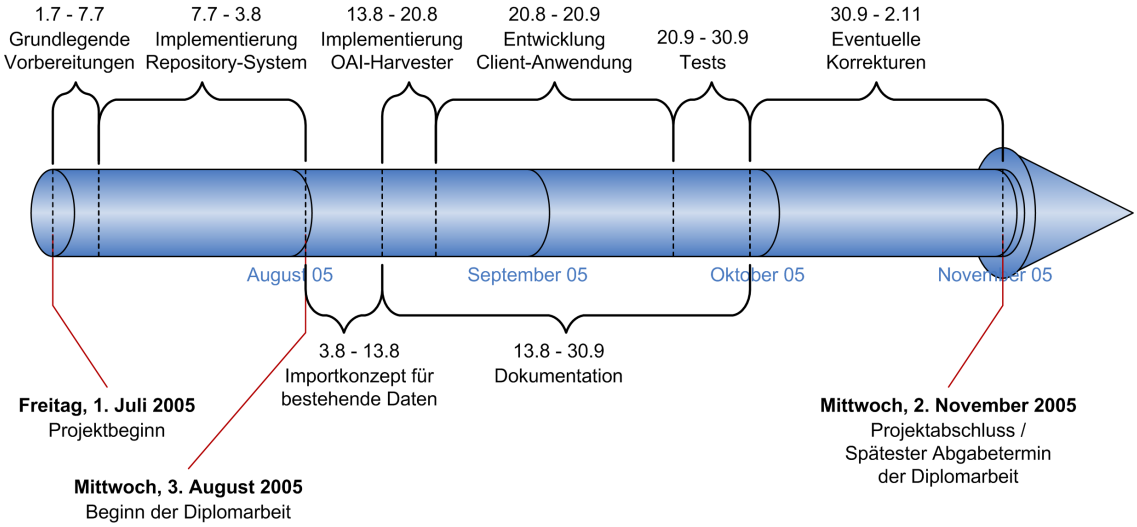


Abbildung 1.3: Projektplan

Kapitel 2

Grundlagen

2.1 Organisation von Informationen

Kleinstes gemeinsames Element ist in der Informatik das Bit. Anreihungen von Bits resultieren letztendlich zu Daten, die in maschinenlesbarer und bearbeitbarer Repräsentation Informationen darstellen sollen. Dessen Organisation wird heutzutage immer wichtiger.

Da Metadaten (eine spezielle Form von Daten bzw. Informationen) eine besondere Bedeutung im Rahmen dieser Diplomarbeit einnehmen, soll dieses Thema in folgendem Abschnitt erläutert werden.

2.1.1 Metadaten

Als Metadaten oder Metainformationen werden allgemein „Daten über Daten“[1] bezeichnet. Das bedeutet, Metadaten sind Informationen die Zusammenstellungen von Daten beschreiben können. Obwohl der Begriff auf den ersten Blick sehr abstrakt klingt, lässt sich die Definition anhand eines einfachen Beispiels veranschaulichen: Handelsübliche Registerkarten einer Bibliothek, auf welchen sich Informationen über die Inhalte eines Buches wiederfinden, enthalten ausschließlich Metadaten. Sie enthalten Merkmale, wie Autor, Erscheinungsjahr, Standort und einige weitere Metainformationen *über* das Buch.

Bei den beschriebenen Daten handelt es sich oft um größere Datensammlungen wie beispielsweise Publikationen. Eine allgemein gültige Unterscheidung zwischen Metadaten und „normalen“ Daten existiert allerdings nicht. So werden auch Angaben von Eigenschaften eines Objektes (zum Beispiel Personennamen) als Metadaten bezeichnet. Während der Begriff „Metadaten“ relativ neu ist, ist sein Prinzip unter

[1] vgl. [SUB01]

anderem jahrhundertelange bibliothekarische Praxis und eigentlich nichts Neuartiges.[2]

Da heutzutage Daten an allen erdenklichen Stellen anfallen (es sei an dieser Stelle beispielsweise auf das Internet verwiesen), stellt sich die Frage nach einer allgemein gültigen Organisationsmöglichkeit von Daten. Das Problem ist das Wiederfinden und die Wiederverwendbarkeit von Daten in größeren Ansammlungen. World Wide Web-Erfinder Tim Berners-Lee, Direktor des World Wide Web Consortium (W3C), hat bereits im Jahre 2001 dazu einen vielversprechenden Lösungsansatz entwickelt.[3] Dieser sieht eine Erweiterung des World Wide Web um maschinenlesbare Daten vor, welche die Semantik[4] der Inhalte formal festlegen (Stichwort: *Semantic Web*). Hintergrund ist, dass zum Einen bessere Kategorisierungsmöglichkeiten zur Verfügung stehen, und zum Anderen mit der Annotation Schlussfolgerungen auf Inhalte möglich werden.

2.1.2 Dublin Core Metadata Element Set

An internationaler Bedeutung hat in den letzten Jahren das Dublin Core Metadata Element Set der Dublin-Core-Metadata-Initiative gewonnen.[5] Es ist mittlerweile das bekannteste Metadatenmodell und besteht aus 15 fest definierten Elementen. Die Initiative setzt sich mit dem Problem „Informationssuche“ schon seit 1995 auseinander und hat im Dezember 2004 ihre zuletzt aktualisierte Version des Metadaten-Element-Sets veröffentlicht.

Im Unterschied zu verschiedenen hochspezialisierten (auch bibliothekarischen) Metadatenschemata (es gibt zahlreiche) ist Dublin Core vergleichsweise einfach angelegt. Gleichzeitig ist es so erweiterbar, dass es auch den Bedürfnissen spezieller Nutzergruppen nach größerer Präzision bei der Indexierung[6] und der Recherche genügen.

Die Metadatenelemente umfassen vorwiegend formalbibliographische und inhaltserschließende Daten bzw. Informationen, die der besseren Indexierung und der Möglichkeit eines gezielten Retrieval verschiedenster Internetressourcen dienen – von einfachen HTML-Dokumenten über Postscript-Dateien bis zu diversen visuellen Formaten. Das Konzept der Metadaten schließt aber grundsätzlich auch Metadatenelemente ein, die sich auf Nutzungs- und Beschaffungsaspekte („terms and conditions“) und die Beziehung zu anderen Objekten („linkage or relationship“) beziehen.[7]

[2] vgl. [Wik05a]
[3] vgl. [BLHL01]
[4] *Semantik*: Sinn und Bedeutung einer Sprache
[5] vgl. [SUB01]
[6] *Indexierung*: Zuordnung von Schlagworten aus einem Schlagwortkatalog bzw. Notationen einer Klassifikation zu einem Dokument zur Erschließung der darin enthaltenen Sachverhalte
[7] vgl. [Cap97]

Folgende 15 Elemente stellt das Set in der aktuellen Version bereit[8]:

Name	Inhalt
`Title`	Der Name der Ressource
`Creator`	Verantwortlicher für den Inhalt
`Subject`	Thema der Ressource
`Description`	Beschreibung des Inhalts
`Publisher`	Verantwortlicher für die Veröffentlichung
`Contributor`	Zur Ressource Beitragende
`Date`	Datum eines Ereignisses im Lebenszyklus der Ressource
`Type`	Beschaffenheit oder Genre des Inhalts
`Format`	Festlegung über die physische oder digitale Erscheinungsform der Ressource
`Identifier`	eindeutige Referenz zur Identifikation der Ressource
`Source`	Referenz zu der Ressource, von der die gegenwärtige abgeleitet ist
`Language`	Verwendete Sprache der Ressource
`Relation`	Referenz zu Ressourcen, die mit der gegenwärtigen in Verbindung stehen
`Coverage`	Ausmaß oder Bereich der Ressource
`Rights`	Informationen zu den Rechten innerhalb oder über die Ressource

Tabelle 2.1: Dublin-Core Metadata Element Set

Detaillierte Informationen zu den Inhalten der einzelnen Elemente lassen sich aus der Referenz zum Dublin Core Metadata Element Set[9] entnehmen. Dort lassen sich auch Ratschläge zum Erstellen sinnvoller Einträge finden.

2.2 Open Access

Seit 2001 suchen weltweite Initiativen nach effizienten Strategien um die Vorteile von Open Access für die Forschung und ihre Institutionen allgemein nutzbar zu machen. Folgender Abschnitt soll einen Überblick darüber geben, was Open Access ist und welche Ziele damit erreicht werden sollen.

2.2.1 Ideologie

Treffend formuliert Katja Mruck, Geschäftsführende Herausgeberin der dreisprachigen Open Access-Zeitschrift *Forum Qualitative Sozialforschung* die Leitgedanken von Open Access:

[8] s. [DCM04]
[9] Dublin Core Metadata Element Set, Version 1.1, s. `http://dublincore.org/documents/dces/`

> „Open Access meint, dass [...] Literatur kostenfrei und öffentlich im Internet zugänglich sein sollte, so dass Interessierte die Volltexte lesen, herunterladen, kopieren, verteilen, drucken, in ihnen suchen, auf sie verweisen und sie auch sonst auf jede denkbare legale Weise benutzen können, ohne finanzielle, gesetzliche oder technische Barrieren jenseits von denen, die mit dem Internet-Zugang selbst verbunden sind. In allen Fragen des Wiederabdrucks und der Verteilung und in allen Fragen des Copyrights überhaupt sollte die einzige Einschränkung darin bestehen, den jeweiligen Autorinnen und Autoren Kontrolle über ihre Arbeit zu belassen und deren Recht zu sichern, dass ihre Arbeit angemessen anerkannt und zitiert wird.“ [Mru02]

2.2.2 Ziele

Open Access soll also den Austausch wissenschaftlicher Arbeiten erleichtern. Da frei zugängliche Publikationen häufiger zitiert werden, steigert Open Access die Sichtbarkeit der Forschungsergebnisse und kann somit das Renommee der Wissenschaftler deutlich erhöhen[10]. Mit der Unterzeichnung der „Berliner Erklärung über den offenen Zugang zu wissenschaftlichem Wissen“[11] vom Oktober 2003 hatte sich neben weiteren Gemeinschaften die HGF[12] dazu verpflichtet, den freien Zugang zu wissenschaftlichen Publikationen im Internet maßgeblich zu unterstützen und zu propagieren.

2.2.3 Open Archives

Grundlegendes Instrument zur Realisierung von Open Access sind Archive, welche die Voraussetzungen für die Nutzung nach den oben genannten Vorgaben schaffen (Open Archives). Open Archives stehen als Synonym für „repositories of scholarly papers with an open architecture“[13], wortwörtlich ins Deutsche übersetzt bedeutet dies: „Architektur-offene Archive für wissenschaftliche Arbeiten“. Diese Archive unterliegen nicht dem „Peer Review“[14], sondern machen ihre Inhalte frei weltweit zugänglich. Sie können unbegutachtete Preprints (Vorabveröffentlichungen vor der Drucklegung), begutachtete Postprints oder beides enthalten.[15]

[10] vgl. [Sie05]
[11] „Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities“, s. `http://www.zim.mpg.de/openaccess-berlin/berlindeclaration.html`
[12] Helmholtz-Gemeinschaft Deutscher Forschungszentren – das Alfred-Wegener-Institut gehört neben weiteren Instituten zum HGF.
[13] vgl. [OAI02]
[14] Bewertung eines Objekts durch unabhängige Gutachter
[15] vgl. [Sub04]

2.2.3.1 Verwendung

Open Archives können von Institutionen wie Universitäten oder Forschungseinrichtungen unterhalten werden. Autoren, bzw. Wissenschaftler, können dort bedenkenlos ihre Preprints einstellen, und eine überwiegende Mehrheit der Zeitschriften in denen publiziert wird gestattet sogar auch das Einstellen der Postprints. Wenn solche Archive sich an das Metadata-Harvesting-Protokoll der Open Archives Initiative (OAI[16]) halten, ist Interoperabilität gegeben und Nutzer können die Inhalte auffinden, ohne zu wissen, welche Archive existieren, wo sie sich befinden und was sie enthalten. Es gibt inzwischen Open Source Software, um solche, dem OAI-Standard entsprechenden Archive, einzurichten. Weltweit gibt es bereits eine breite Akzeptanz für die Nutzung dieser Software. Bekannte Systeme sind neben zahlreichen anderen *DSpace*[17], *EPrints*[18] und *Fedora*[19].

2.2.4 OAI Protocol for Metadata Harvesting (OAI-PMH)

Realisiert wird die offene Architektur von Open Archives durch deren Schnittstellen. Das bereits seit 2000 existierende *Protocol for Metadata Harvesting* (OAI-PMH) bietet für die Integration von Open Archives Möglichkeiten dazu, elektronische Publikationen im Internet durch das Bereitstellen passender Schnittstellen auffindbar zu machen[20]. Auf dieses Thema wird später in Abschnitt 3.1.1 ab Seite 30ff. detailliert eingegangen.

2.3 Web Services

Doch es gibt auch alternative Ansätze. Web Services haben in den letzten Jahren immer mehr an Bedeutung gewonnen. Als eine der wichtigen neuen Technologien sind sie allerdings noch nicht weit in die tägliche Praxis von Unternehmen und Institutionen vorgedrungen.

2.3.1 Ziel

Ursprünglich lag die Verwendung der Web Service-Technologie hauptsächlich im E-Business-Bereich. Mit Web Services sollten elektronische Marktplätze aufgebaut und Unternehmen über ihre Grenzen hinweg vernetzt werden, um unternehmensübergreifende Anwendungen zu schaffen. Mittlerweile finden Web Services ihre Anwendung auch vereinzelt im Bereich der unternehmensinternen Anwendungsintegration.

[16] s. `http://www.openarchives.org`
[17] „A groundbreaking digital repository system“ – s. `http://www.dspace.org`
[18] „Self Archiving and Open Access Eprint Archives“ – s. `http://www.eprints.org`
[19] „General purpose repository service“ – s. `http://www.fedora.info`
[20] vgl. [OAI04]

Ziel ist es, auf einfache Art und Weise die häufig verschiedenen Anwendungen innerhalb eines Unternehmens flexibel zu koppeln. Auch in dieser Arbeit soll unter der Verwendung von Web Services eine flexible, lose gekoppelte Anwendung konzeptioniert und prototypisch umgesetzt werden.

2.3.2 Ursprung

Dieser Traum von uneingeschränkter Interoperabilität ist jedoch schon älter. Zuletzt wurde mit der *Common Object Request Broker Architecture* (CORBA), entwickelt von der Object Management Group (OMG), eine Lösung versucht. Davor war das *Distributed Component Object Model* (DCOM), eine Erweiterung des DCOM Microsoft-Objektmodells COM für verteilte Umgebungen, eine ebensolche Bemühung. Der im JDK enthaltene Mechanismus zur Programmierung verteilter Objekte in Java, *Java Remote Method Invocation* (RMI), ist ein weiterer Lösungsansatz.

Im Gegensatz zu den Vorläufern verfolgen Web Services jedoch einige neue Grundsätze: Explizit lose Kopplung zwischen *allen* Beteiligten, Dezentralität, Verwendung offener, teilweise bereits existierender Standards, Nutzung einfacher und leicht umzusetzender Protokolle. Durch die Verwendung bereits breit akzeptierter Standards, wie z.B. HTTP, sind Web Services leicht in einem heterogenen Umfeld einsetzbar. Die Verwendung von CORBA, Java RMI oder DCOM setzt dagegen zumeist auch die Verwendung eines bestimmten Betriebssystems (Microsoft bei DCOM) und/oder einer bestimmten Programmiersprache (Java bei Java RMI) voraus. Die plattform- und weitestgehend sprachunabhängige CORBA kann auf allen entsprechend kompatiblen Systemen verwendet werden. CORBA ist jedoch (wie auch DCOM) sehr komplex, und eine entsprechende Systemanpassung ist daher oft sehr aufwändig. Der Anpassungsaufwand für die Verwendung von Web Services ist dagegen eher gering. Auch der Einarbeitungsaufwand in diese Technologie ist aufgrund der Verwendung bekannter Standards bei weitem nicht so umfangreich wie beispielsweise für CORBA.

2.3.3 Technologie

Bei Web Services handelt es sich um eine nachrichtenorientierte Technologie. Über Netzwerke werden Nachrichten gesendet und empfangen. Servicebenutzer senden Nachrichten an Serviceanbieter, die mit den Ergebnissen der Anfrage antworten. Dazu werden die ausgetauschten Nachrichten unter Verwendung von standardisierten Protokollen mit beispielsweise HTTP oder FTP übertragen.[21]

Ein Web Service ist somit die Schnittstelle zwischen einer Anwendung und einem (meist entfernten) Servicebenutzer, der die Anwendungsfunktionen nutzen möchte.

[21] vgl. [Kre04]

Dabei ist es vollkommen gleichgültig, welche Programmiersprache der Servicebenutzer verwendet und auf welcher Plattform gearbeitet wird. Aus diesem Grund spricht man auch von einer *Abstraktionsschicht*, die durch Web Services realisiert wird. Einzige Voraussetzung seitens des Servicebenutzers bzw. des Serviceanbieters ist, dass die Standardtechnologien des Internets verwendbar sind. Die im Weiteren beschriebene Architektur von Web Services verdeutlicht, wie dies möglich ist.

2.3.3.1 Extensible Markup Language (XML)

XML, eine Teilmenge der Beschreibungssprache SGML[22], ist seit Februar 1998 ein Standard des W3C und wurde dazu entwickelt plattformübergreifend Daten zu strukturieren, zu beschreiben und gleichzeitig als Datenträger zu fungieren, der Software- und Hardware-unabhängig genutzt werden kann. Sie ist keinesfalls ein Ersatz für HTML oder XHTML, wie oft angenommen wird – HTML und XHTML verfolgen gänzlich andere Ziele, wenngleich die Ähnlichkeit mit XML vorhanden ist.

Ähnlich einer CSV[23]-Datei enthält es Daten, die jedoch in eine sich selbst-beschreibende Baum-Struktur gebettet werden. Dazu sind die Datensätze mit so genannten *Tags* verbunden, welche Informationen über den Inhalt enthalten und die Daten somit beschreiben können. Sie gilt damit als *selbst-beschreibend*. Zusätzlich kann eine XML-Datei durch eine *Document Type Definition* (DTD) oder ein *XML-Schema* einer bestimmten Struktur unterliegen, die innerhalb eines XML-Dokumentes eingehalten werden muss. Diese Strukturinformationen sind ebenfalls selbst-beschreibend.

Folgendes Beispiel zeigt den Aufbau eines einfach-gehaltenen XML-Dokuments:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <publikation>
3   <titel>XML, der universale Standard</titel>
4   <autoren>
5    <erstautor>Max Mustermann</erstautor>
6    <coautor>Peter Petersen</coautor>
7   </autoren>
8   <typ>Diplomarbeit</typ>
9   <jahr>2005</jahr>
10 </publikation>
```

Quelltext 2.1: Beispiel: XML-Dokument

Ein XML-Dokument muss genau ein Element in der obersten Ebene enthalten. Unterhalb dieses Elements können weitere Unter-Elemente verschachtelt werden. Diese XML-Tags sind nicht vordefiniert, beim Erstellen eines XML-Dokumentes müssen die Tags nach einem vorgegeben Schema vergeben werden oder selbst „erfunden“

[22] Standard Generalized Markup Language: Eine Beschreibungssprache, mit deren Hilfe verschiedene Auszeichnungssprachen (engl. *markup languages*) für Dokumente definiert werden können.
[23] Character Separated Values: Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten

werden. Allerdings sind die Freiheiten beim Entwerfen von XML-Dokumenten durch Regeln beschränkt, die es einzuhalten gilt[24]:

- XML-Dokumente müssen *wohlgeformt* (engl. *well-formed*) sein – Es ist wohlgeformt, wenn es sämtliche Regeln für XML einhält, was z.B. das Verschachteln von Elementen betrifft.
- XML-Dokumente müssen *gültig* (engl. *valid*) sein – Soll XML für den Datenaustausch verwendet werden, ist es aus Gründen der Interoperabilität von Vorteil, wenn das Format mittels einer Grammatik (z.B. einer DTD oder einem XML-Schema) definiert ist. Ein XML-Dokument, welches wohlgeformt ist und ein durch eine Grammatik beschriebenes Format einhält, heißt *gültig*.

In der heutigen Praxis verarbeiten Systeme, Anwendungen und Datenbanken fast ausschließlich Daten in verschiedenen, zueinander inkompatiblen Formaten. XML versucht nun eine Schnittstelle für all die verschiedenen Formate zu bieten – Generalisierte Formate, die durch Maschinen wie auch Menschen lesbar sind, werden durch die Verwendung von XML möglich. Ein solches Format zum Transport von Daten, welches im Zuge von Web Services genutzt wird, ist SOAP.

2.3.3.2 Simple Object Access Protocol (SOAP)

Ein Akronym als Programmname wählen liegt seit langem im Trend und so spielte dieses Motiv bei der Namensfindung von SOAP wohl auch eine Rolle. Dennoch hat das Simple Object Access Protocol wenig mit laugenartigen Substanzen zu tun, vielmehr dient SOAP zum „Verpacken“ von Nachrichten. Mittlerweile gilt das XML-basierte Protokoll beim Einsatz von XML-RPC[25] sogar als Standardlösung. SOAP „verpackt“ Nachrichten in ein für alle Beteiligten verständliches Format.

SOAP spezifiziert im Rahmen von Web Services das Nachrichtenformat zwischen Client und Server. Durch die weit verbreitete Sprache XML wird sichergestellt, dass SOAP plattformunabhängig agieren kann. Zusätzlich ist gewährleistet, dass dieses Protokoll unabhängig der Programmiersprache, des verwendeten Netzwerks, sowie der Transportschicht genutzt werden kann. Da es ein rein textbasiertes Übertragungsprotokoll ist, hat es keine Probleme mit Firewalls oder Proxies und eignet sich deshalb dazu, beliebige Funktionen im Internet bereitzustellen – Ideal zur Realisierung von Web Services.

[24] vgl. [Wik05f]
[25] XML Remote Procedure Call

Eine SOAP-Nachricht wird durch einen *SOAP-Envelope* realisiert. Er besteht optional aus einem Header und zwingend aus einem Body. Letzterer enthält den Inhalt der eigentlichen Nachricht, also die „verpackten“ Daten, welche in applikationsspezifischem Format vorliegen. Die Angaben im Header des Envelope steuern die Verarbeitung und sind ebenfalls applikationsspezifisch.

Abbildung 2.1 macht den Aufbau eines SOAP-Envelope schematisiert deutlich.

Die Praxis zeigt folgendes Listing (aus [GHM+03, Section 1.4]). In diesem Fall wird ein Header verwendet, der applikationsspezifische Steuerungsdaten enthält. Das Body-Element enthält die zu transportierenden Daten:

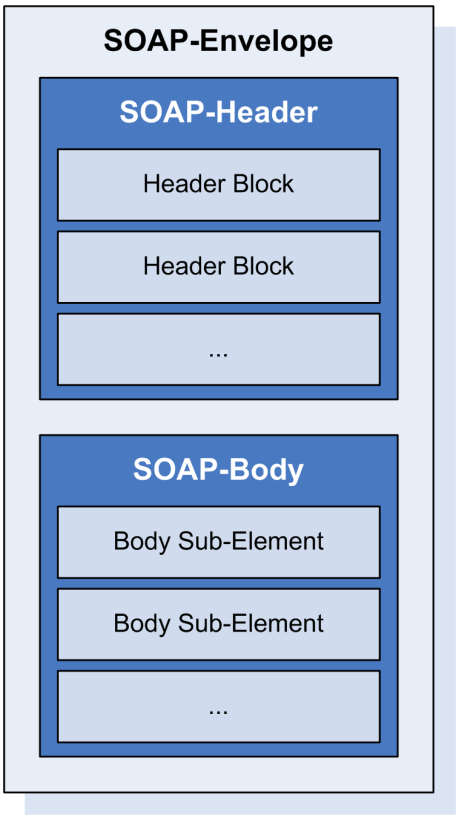


Abbildung 2.1: SOAP Envelope

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Quelltext 2.2: Beispiel SOAP-Envelope

2.3.3.3 Web Service Description Language (WSDL)

Damit der Servicebenutzer konkrete Dienste kontaktieren und verwenden kann, müssen diese verständlich beschrieben sein. Dazu gehören die Angaben von Adresse, Formaten, Operationen, Parametern und Datentypen. Die WSDL (ebenfalls XML-

basierend) gilt zur Zeit als Standard[26] für eine solche Beschreibung. Weitere, weniger verbreitete Ansätze sind das *Resource Description Framework* (RDF) des W3C und die *DARPA Agent Markup Language* (DAML). Anhand dieser Beschreibung und mit Hilfe der Funktionen in der Beschreibungsschicht wird dem System bekannt gemacht, welche Protokolle für die unteren Schichten (Netzwerk, Transport und Verpackung) verwendet werden.

Ein WSDL-Dokument kann man sich als eine Ansammlung von Definitionen vorstellen, die einen Web Service mit all seinen Eigenschaften festlegen. Diese Festlegungen sind in zwei Gruppen aufteilbar:

- *Abstrakte Definitionen*: In diesen werden Typen, Nachrichten und Sende- bzw. Empfangsoperationen definiert.
- *Konkrete Definitionen*: In diesen werden die oben genannten Definitionen einer Bindung und einer Netzwerkadresse zugeordnet.

Zu den abstrakten Definitionen gehören die Elemente *types*, *message* und *portType*. Zu den konkreten Definitionen werden die Elemente *binding* und *service* gezählt.

Eine WSDL-Datei enthält immer ein äußeres Element. Es heißt *definitions* und enthält die Elemente *types*, *message*, *portType*, *binding* und *service*. Diese wiederum enthalten weitere Unter-Elemente.

Im Element *types* befindet sich das Element *schema*, im Element *message* das Element *part*. Die Elemente *portType* und *binding* haben jeweils das Unter-Element *operation* und das Element *service* enthält das Element *port*.

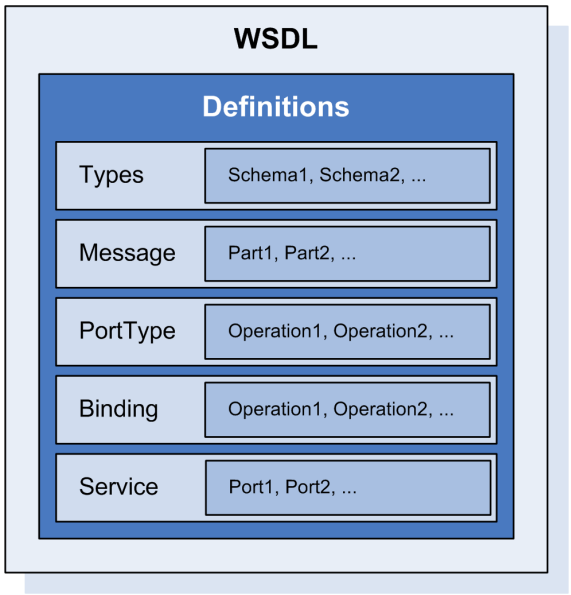


Abbildung 2.2: Schema eines WSDL-Dokuments

Abbildung 2.3 zeigt die Verknüpfungen der Elemente eines WSDL-Dokuments untereinander.

[26] vgl. [CGMW03]

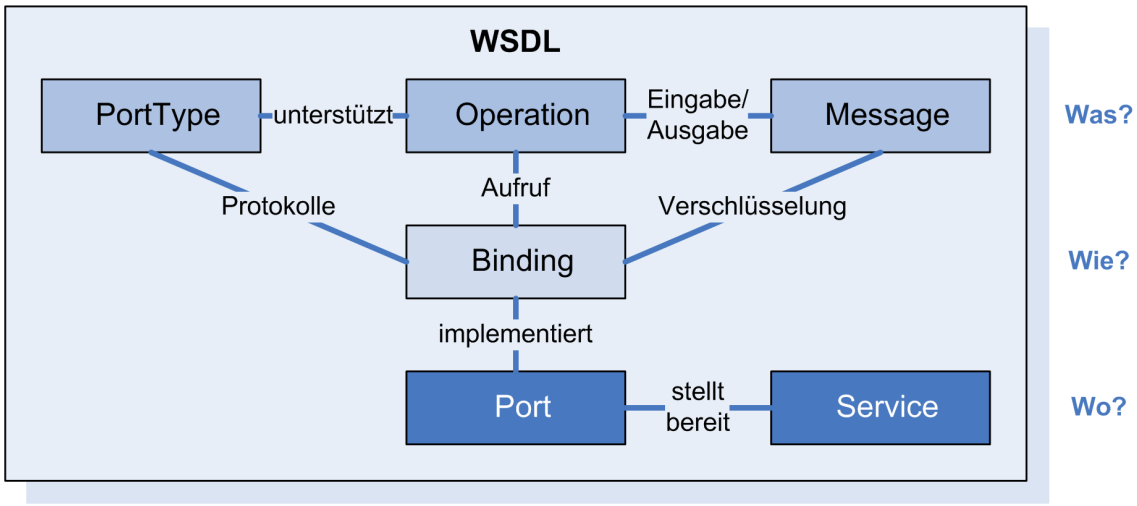


Abbildung 2.3: Verknüpfungen der Typen, vgl. [KL04]

Diese komplexe Strukturierung wird anhand des folgenden Listings eines WSDL-Dokuments mit entsprechenden Inhalten verständlicher. Hier wird ein Web Service beschrieben, der die Funktionalität bereitstellt, die einfache Rechenoperation *Addition* durchzuführen; im Anschluss folgt eine detaillierte Beschreibung:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions
3     targetNamespace="http://localhost/axis/Calc.jws"
4     xmlns="http://schemas.xmlsoap.org/wsdl/"
5     xmlns:apachesoap="http://xml.apache.org/xml-soap"
6     xmlns:impl="http://localhost/axis/Calc.jws"
7     xmlns:intf="http://localhost/axis/Calc.jws"
8     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
9     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
10    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
11    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
12   <wsdl:portType name="Calc">
13    <wsdl:operation name="add" parameterOrder="a b">
14     <wsdl:input message="impl:addRequest" name="addRequest"/>
15     <wsdl:output message="impl:addResponse" name="addResponse"/>
16    </wsdl:operation>
17   </wsdl:portType>
18   <wsdl:message name="addRequest">
19    <wsdl:part name="a" type="xsd:int"/>
20    <wsdl:part name="b" type="xsd:int"/>
21   </wsdl:message>
22   <wsdl:message name="addResponse">
23    <wsdl:part name="addReturn" type="xsd:int"/>
24   </wsdl:message>
25   <wsdl:binding name="CalcSoapBinding" type="impl:Calc">
26    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
27    <wsdl:operation name="add">
28     <wsdlsoap:operation soapAction=""/>
29     <wsdl:input name="addRequest">
30      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
            DefaultNamespace" use="encoded"/>
31     </wsdl:input>
32     <wsdl:output name="addResponse">
```

```
33      <wdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://
            localhost/axis/Calc.jws" use="encoded"/>
34    </wsdl:output>
35   </wsdl:operation>
36  </wsdl:binding>
37  <wsdl:service name="CalcService">
38   <wsdl:port binding="impl:CalcSoapBinding" name="Calc">
39    <wsdlsoap:address location="http://localhost/axis/Calc.jws"/>
40   </wsdl:port>
41  </wsdl:service>
42 </wsdl:definitions>
```

Quelltext 2.3: Beispiel WSDL

Ein WSDL-Dokument lässt sich einfacher verstehen, wenn am unteren Ende mit dem Lesen begonnen wird. Der Web Service aus dem obigen Beispiel heißt also `CalcService` und ist unter `http://localhost/axis/Calc.jws` via SOAP erreichbar. Er nutzt zur Bereitstellung der Funktionen den abstrakt definierten Port `Calc` der aus nur einer Funktion besteht: `add`. Die Funktion besitzt zudem zwei Messages: für die Anfrage an den Service `addRequest` und für die Antwort mit Ergebnis den `addResponse`. Erstere erfordert zwei Parameter (die zu addierenden Integer-Werte), die Antwort besteht aus Einem (Integer-Wert des Ergebnisses der Rechenoperation). Durch die angegebenen Namespaces ist der Service und dessen Operationen, Funktionen und Datentypen nach den jeweiligen Schemata festgelegt und durch einen Web Service-Client validierbar. Dieser hat nun alle erforderlichen Information zur Nutzung des Web Services durch das WSDL-Dokument beisammen und kann diesen aufrufen – beispielsweise durch die Integration in einer eigenen Anwendung.

2.3.4 Kombination der Technologien

Eine Web Service-Architektur besteht typischerweise aus drei Beteiligten: Service Anbieter, Service Registrierung und Service Benutzer. Die folgende Abbildung macht das Zusammenwirken unter Berücksichtigung der verwendeten Protokolle deutlich:

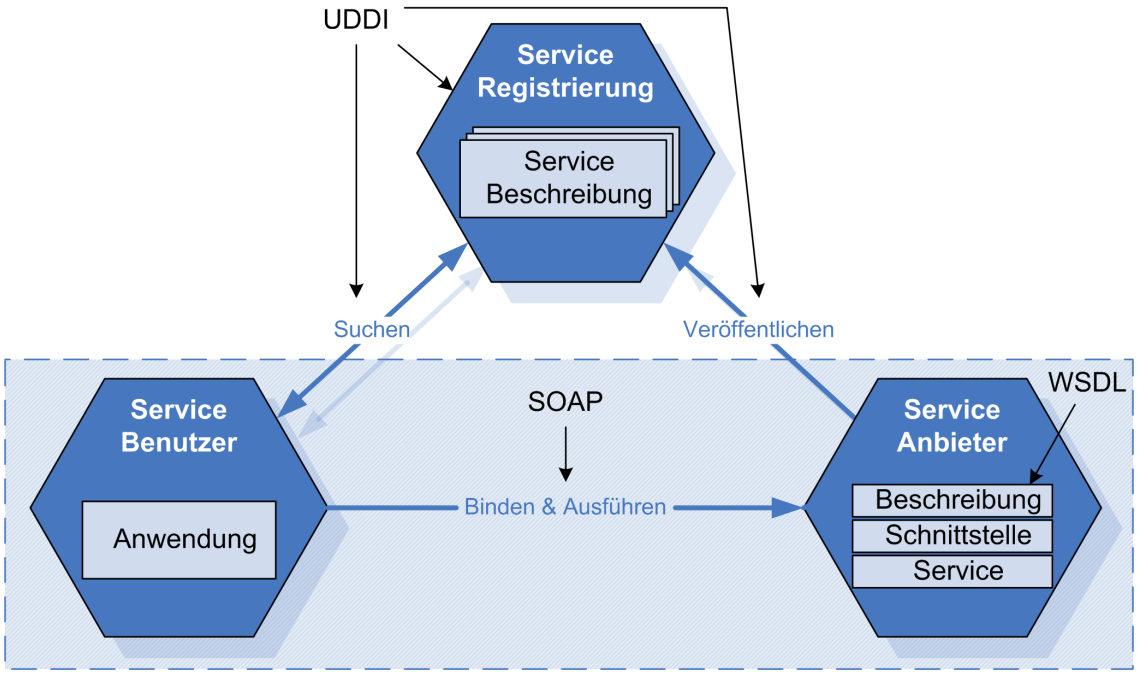


Abbildung 2.4: Web Services (vgl. [Kre04])

Der Service Anbieter, der einen Web Service anbieten will, legt dazu auf seinem Server ein WSDL-Dokument ab. Um auf seinen Web Service aufmerksam zu machen, veröffentlicht er danach seinen Web Service bei einer Service Registrierung (zumeist durch UDDI[27] realisiert), indem er die Adresse des WSDL-Dokuments und weitere Informationen über den Web Service hinterlegt. Die Service Registrierung soll somit als Vermittler zwischen Service Anbieter und Service Benutzer agieren.

Der Service Benutzer wendet sich zuerst an die Service Registrierung, um einen gewünschten Web Service zu suchen. Hat er einen Web Service gefunden kann er anhand der angegebenen Adresse das WSDL-Dokument ausfindig machen und es auswerten. Die Definitionen in dem WSDL-Dokument spezifizieren das Protokoll, die Adresse und andere Dinge, die es ihm ermöglichen, mit dem Service Anbieter in Verbindung zu treten, um den gewünschten Web Service zu benutzen.

Die Aufgabenstellung dieser Diplomarbeit berührt ausschließlich die Seite des Service Anbieters und des Service Benutzers. Die Rolle der Service Registrierung bleibt außer Acht.

[27] UDDI – „Universal Description, Discovery and Integration“: Ein Protokoll welches zum Auffinden von Web Services verwendet wird. UDDI wird zwar immer Zuge von Web Services genannt, spielt aber in dieser Arbeit keine Rolle. UDDI ist nicht zwingend erforderlich um mit Web Services zu arbeiten oder sie zu implementieren.

2.4 Repositories

Repositories lassen sich im Allgemeinen mit Datenbanken vergleichen. Sie stellen Mechanismen zur Verfügung, die es erlauben Daten zu speichern, wiederzufinden und zu verändern. Folgende Definition gibt eine prägnante Aussage darüber, durch welche besonderen Eigenschaften ein Repository charakterisiert wird.

2.4.1 Definition

> „A repository is a central place where data is stored and maintained. A repository can be a place where multiple databases or files are located for distribution over a network, or a repository can be a location that is directly accessible to the user without having to travel across a network.“ [28]

2.4.2 Besondere Eigenschaften von Repositories

Bei Repositories wird im Gegensatz zu klassischen Datenbanken nicht unterschieden, um welche Art von Daten es sich handelt. Prinzipiell lässt sich in einem Repository jegliche Art von Information speichern, solange diese digitalisierbar ist. Speicherbar werden durch den Einsatz eines Repository beispielsweise Artikel, Reporte, Bücher, aber auch Audio- oder Video-Sequenzen. Die Klassifizierung der in einem Repository enthaltenen Daten geschieht anhand von Metadaten, beispielsweise dem Dublin Core-Metadatenschema, welche die Eigenschaften von Objekten festhalten. Anhand dessen können später Rückschlüsse auf deren Inhalt gezogen werden. Der Vorteil solcher Metadaten-Repositories liegt darin, dass diese Systeme ohne Programmieraufwand flexibel auf Änderungen reagieren können.

Weiterhin charakteristisch für ein Repository ist ein integriertes Versionsmanagement. Diese Technologie erlaubt es, beliebige Zustände von gespeicherten Objekten bereitzuhalten. Jede Änderung am Datenbestand wird protokolliert und kann nachverfolgt oder rückgängig gemacht werden.

Ein wiederum optionales Charakteristikum stellt ein integriertes Rechtemanagement dar, welches die Regelung von Zugriffsberechtigungen auf ein Repository übernehmen kann. Idealerweise ist in dieses Rechtemanagement eine zusätzliches Rollenbasiertes Workflow-System implementiert, durch welches Bearbeitungsvorgänge für die Datenbestände global festlegbar sind.

[28] aus [Wik05c]

2.4.3 Anforderungen

Allerdings sind digitale Inhalte nicht nur digitalisierbare, einfache Textdokumente. Mit einer tiefgehenden Betrachtungsweise kommt man zu dem Schluss, dass Inhalte durchaus komplexer sein können, als auf dem ersten Blick angenommen. Eine Klassifizierung von digitalem Inhalt teilt Objekte in zwei Gruppen:

- *Konventionelle Objekte:* z.B. Bücher und andere text-basierte Inhalte, Bilder
- *Komplexe, Assoziative, Dynamische Objekte:* z.B. Video, Audio, Datensätze mit verknüpften Referenzen

Unter Berücksichtigung der Komplexität von digitalen Inhalten müssen sich die Entwickler von Repositories folgende Schlüsselfragen stellen[29]:

- Wie können Clients in einer einfachen, aber interoperablen Art und Weise mit Sammlungen verschiedenartiger, komplexer Objekte interagieren?
- Wie sollten komplexe Objekte allgemein gestaltet werden, wenn sie generisch und gleichzeitig typenspezifisch sein sollen?
- Wie können verschiedene Präsentationen oder Transformationen von digitalen Inhalten realisiert werden?
- Wie können feinkörnige, objektbezogene Zugriffsrechte implementiert werden?
- Wie kann eine Langzeitarchivierung und dessen Management erreicht werden?

All jene Fragen hat sich auch das Entwickler-Team um *Fedora* gestellt. Im Folgenden soll kurz auf Architektur und Integration von Diensten zur Erfüllung der Anforderungen an ein Repository-System eingegangen werden, da jenes im Zuge dieser Ausarbeitung zum Einsatz kommt.

2.4.4 Das Repository-System „Fedora“

Fedora steht für die Abkürzung „Flexible Extensible Digital Object Repository Architecture“[30]. Es ist ein flexibles Repository-System, welches für verschiedene Anwendungsgebiete geeignet ist. Beispielsweise lassen sich damit komplexe Anwendungen wie *Digital Asset Management*, *Institutionelle Repositories*, *Digitalarchive*, *Content-Management-Systeme*, *digitale Bibliotheken* oder *Systeme zur Veröffentlichung von Publikationen* realisieren. Fedora ist Open-Source-Software und mit der *Mozilla Public License*[31] lizenziert.

[29] vgl. [Fed05, Section 2]
[30] vgl. [Fed05, Section 1]
[31] vgl. [Moz04]

2.4.4.1 Architektur

Ein Fedora Repository speichert digitale Objekte und verwaltet diese. Des Weiteren stellt es Interaktionsmöglichkeiten zur Verfügung, mit dessen ein Zugriff auf digitale Objekte möglich ist.

Das folgende Diagramm (Abb. 2.5) zeigt die vereinfachte Struktur des Fedora Repository-Kerns. Dem Benutzer werden mehrere Möglichkeiten zur Verfügung gestellt, um auf die Inhalte zuzugreifen:

- durch *Anwendungen auf Client-Seite*,
- durch *Web Browser*,
- mittels *Batch Programmen* (Stapelverarbeitungen) oder
- durch *Server Applikationen*.

Diese Anwendungen können auf die Inhalte des Repository mittels der integrierten vier möglichen API's zugreifen. Die API's *Manage*, *Access* und *Search* sind durch Web Services realisiert und über HTTP oder SOAP zu erreichen. Auf das *OAI-Provider*-API kann via HTTP zugegriffen werden.

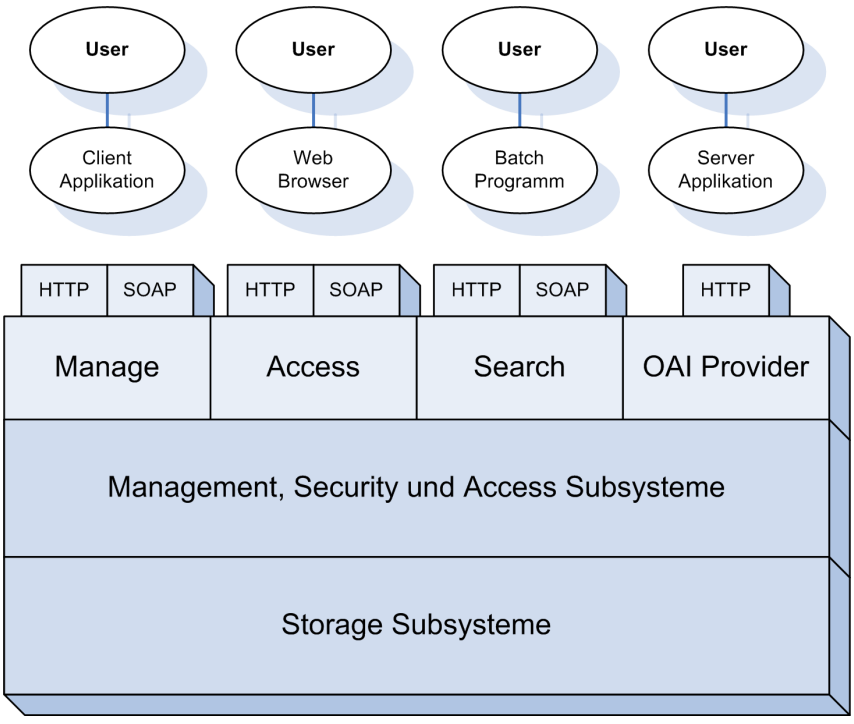


Abbildung 2.5: Vereinfachte Architektur des Fedora Repository (vgl. [Fed05, Section 4])

Abbildung 2.6 gibt einen weiteren Einblick in die Interaktionsmöglichkeiten des Fedora Repository.

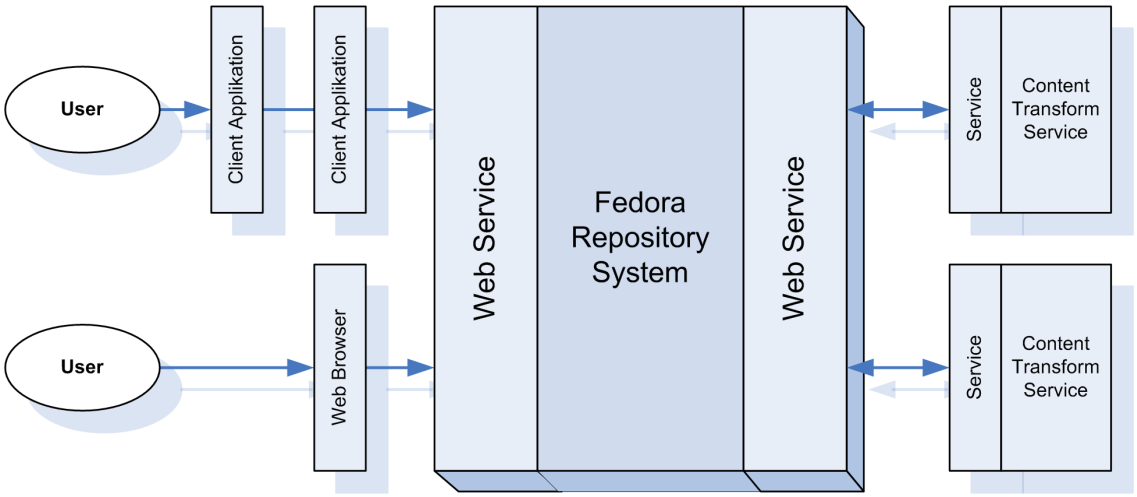


Abbildung 2.6: Interaktionsmöglichkeiten mittels Web Services (vgl. [Fed05, Section 4])

Generell stellen Benutzer Anfragen an das Repository wie beispielsweise zum Einfügen von Objekten (*ingest objects*), zum Durchsuchen der Datenbestände des Repository (*search repository*) oder zum Anzeigen von Objekten (*access objects*) via Client-Anwendung oder Web Browser. Die Anfragen werden durch Web Services am Frontend des Systems an dessen Kern vermittelt. Das Backend kann ebenfalls aus einer Reihe von Web Services bestehen, die zusätzliche, selbst-definierbare Funktionen ausführen können, die der Benutzer mit seiner Anfrage angefordert hat. Die verarbeiteten Daten werden, nachdem alle erforderlichen Transformationen durchgeführt sind, als Ergebnis der Anfrage an den Benutzer zurückgesendet. Dies geschieht ebenfalls durch die Web Services am Frontend des Systems.

Es ist wichtig zu wissen, dass Benutzeranfragen ausschließlich über die integrierten API's abgewickelt werden. Obwohl es so scheint, als greife man direkt auf Objekte selbst zu, interagiert man stattdessen mit den Methoden des Repository, welche den Zugriff auf Objekte und dessen verknüpfte Funktionen organisieren und bereitstellen.

2.4.5 Institutionelle Repositories

Institutionelle Repositories enthalten im Wesentlichen den produktiven Output wissenschaftlicher Forschungsergebnisse. Je nach Größe einer Institution und Umfang der Forschungen entsteht eine bestimmte Menge digitaler Inhalte, welche durch ein solches Repository verwaltet werden muss. Die klassischen Verfahren zur Datenhaltung werden von Repositories fortwährend abgelöst, da jene digitale Inhalte flexibler verwalten und bereitstellen können. Die Vorteile überwiegen auf Betreiberseite, wie auch auf der Seite der Repository-Benutzer.

Für wissenschaftliche Forschungseinrichtungen kann die Nutzung von Repositories von großem Vorteil sein, denn dessen oben genannte Konzeptionierung kann unterstützend in den Prozess zur (Langzeit-)Archivierung von großen Datenmengen, wie beispielsweise Publikationen, eingreifen. Das Aufgabenspektrum dieser Diplomarbeit ist aus diesem Grund darauf ausgerichtet, eine prototypische Umgebung eines Institutionellen Repository zu erschaffen und dieses auf eine eventuelle, zukünftige Nutzung vorzubereiten.

Daraus ergeben sich folgende Leitgedanken, die bei der Erarbeitung der Ergebnisse dieser Diplomarbeit berücksichtigt werden sollen:

- Digitale Inhalte sollen nach den Richtlinien von Open Access zugänglich gemacht werden.
- Die Ergebnisse der Entwicklungen dieser Arbeit sollen auf eine Langzeitarchivierung ausgerichtet sein.

2.5 Entwicklungsumgebung

Folgender Abschnitt erläutert die verwendeten Anwendungen und Werkzeuge, die während der zu erarbeiteten Diplomarbeit genutzt wurden oder zur Verwendung gekommen sind. Unter Anderem wird grundlegend auf Funktionsweise und die betreffende Stelle innerhalb der Diplomarbeit eingegangen.

2.5.1 Apache HTTP Server

Auf Entwicklungsseite sowie auf Produktivseite wurde der Apache-Webserver (Apache HTTP Server) der Apache Software Foundation eingesetzt. Er ist der meist verwendete Webserver weltweit[32].

Neben Unix und Linux läuft der Apache auch auf Windows, NetWare sowie einer Vielzahl weiterer Betriebssysteme. In der aktuellen Version 2.0 wurde die Stabilität und Geschwindigkeit des Servers erheblich verbessert: Die Bibliothek *Apache Portable Runtime* (APR) stellt eine Verallgemeinerung wichtiger Systemaufrufe zur Verfügung, so dass die individuellen Stärken des jeweiligen Betriebssystems ausgenutzt werden können. Hinzu kommen verschiedene *Multiprocessing-Module* (MPM), die je nach Plattform unterschiedliche Lösungen für die gleichzeitige Bedienung mehrerer Client-Anfragen anbieten, was besonders interessant für die Lösung der Aufgabenstellung dieser Arbeit ist – aber dazu später mehr.

Der Apache HTTP Server ist modular aufgebaut: Durch entsprechende Module kann er beispielsweise die Kommunikation zwischen Browser und Webserver verschlüsseln, als Proxy-Server eingesetzt werden oder komplexe Manipulationen von

[32] vgl. [Net05]

HTTP-Headern und URLs durchführen. Er bietet neben der Darstellung von Hypertext[33] mittels HTML (vgl. Abschnitt 2.5.2) die Möglichkeit, mittels serverseitiger Skriptsprachen Webseiten dynamisch zu erstellen. Häufig verwendete Skriptsprachen sind PHP oder Perl. Diese sind kein Bestandteil des Webservers, sondern müssen ebenfalls entweder als Module eingebunden werden oder über die CGI-Schnittstelle angesprochen werden. Allerdings sind die Module zumeist ohne aufwändige Konfiguration integrierbar.

Der Apache HTTP Server ist, wie alle Produkte der Apache Software Foundation, frei als Open Source unter der Apache-Lizenz verfügbar. Die in dieser Arbeit genutzte Version ist die 2.0.54.

2.5.2 Extensible HyperText Markup Language (XHTML)

Die XHTML ist ebenso wie ihr Vorgänger HTML ein Dokumentenformat zur Auszeichnung von Text im World Wide Web. HTML gibt es schon seit 1998 und wurde einst von Tim Berners-Lee am CERN in Genf entwickelt. Wie die XML basiert sie auf der Metasprache SGML. Die vom W3C weiterentwickelte Sprache XHTML unterscheidet sich von HTML allerdings darin, dass die strengere und einfacher zu parsende SGML-Teilmenge XML als Sprachgrundlage verwendet wird. XHTML-Dokumente entsprechen also den Syntaxregeln von XML[34].

Da sich XHTML als Nachfolger einer Sprache mit zahlreichen Dialekten (gemeint ist HTML) näher an einen weit verbreiteten Standard hält, ist es einleuchtend eine solche Sprache für die Realisierung von Projekten zu verwenden, die auf soliden, zukunftssicheren Technologien bestehen sollen. Aus diesem Grund soll bei der Entwicklung der Ergebnisse dieser Arbeit die XHTML in der Version 1.1 als Grundlage zur Ausgabestrukturierung der Benutzerschnittstellen des Prototyps verwendet werden.

2.5.3 Cascading Style Sheets (CSS)

Durch die Textauszeichnungssprachen HTML und XHTML werden logische Bestandteile eines Dokumentes ausgezeichnet. Sie dienen also lediglich dazu, Strukturen festzulegen. Daraus folgt, dass Schrift, Farb- Größen- und Positionsdefinitionen nicht deren Aufgabe ist. Cascading Style Sheets (CSS) bezeichnen eine Formatierungssprache, die es Web-Autoren und -Benutzern gestattet, Formatierungen (zum Beispiel Schriften, Abstände, Rahmen, Hintergrundfarben, Positionierung) von strukturierten Dokumenten, beispielsweise von XHTML-Dokumenten, durchzuführen. Mit der Trennung der Präsentation eines Dokuments vom Inhalt des Do-

[33] Nichtlineare Organisation von textueller Information
[34] vgl. [Wik05e]

kuments vereinfacht CSS die Erfassung von Web-Dokumenten und die Verwaltung von Webseiten[35].

CSS ist wie HTML selbst ein softwareunabhängiger Textcode und wird vom W3C normiert. Mittlerweile gibt es zwei offizielle Sprachversionen von Cascading Style Sheets: CSS 1.0 aus dem Jahre 1996 und CSS 2.0 von 1998. Die Versionen 2.1 und 3.0 sind in Arbeit. Zur Festlegung des Layouts der Benutzerschnittstellen der zu entwickelnden Anwendung dieser Diplomarbeit soll CSS in der Version 2.0 zum Einsatz kommen.

2.5.4 Hypertext Preprocessor (PHP)

PHP steht als Abkürzung für *„PHP: Hypertext Preprocessor“*. Seit der Einführung der Sprache als Abspaltung von *Perl* im Jahre 1995 liegt PHP mittlerweile in der Version 5 vor. Sie ist eine weit verbreitete Open-Source-Skriptsprache und hat sich auf die Entwicklung von Internetapplikationen spezialisiert. Neben der Ausführung von PHP-Quellcode auf Kommandozeile lässt sich PHP in HTML einbinden, Voraussetzung dafür ist die Integration in den Web Server. Die Syntax erinnert an C, Java und Perl und ist einfach zu erlernen. Das Hauptziel dieser Sprache ist es, Webentwicklern die Möglichkeit zu geben, unkompliziert dynamische Webseiten zu erzeugen.[36]

2.5.4.1 Funktionsweise

Die Funktionsweise von PHP und dessen Integration in einen Web Server ist im Folgenden dargestellt. In diesem Beispiel greift ein Benutzer per Web Browser auf die Datei `beispiel.php` zu, die auf einem entfernten Web Server liegt:

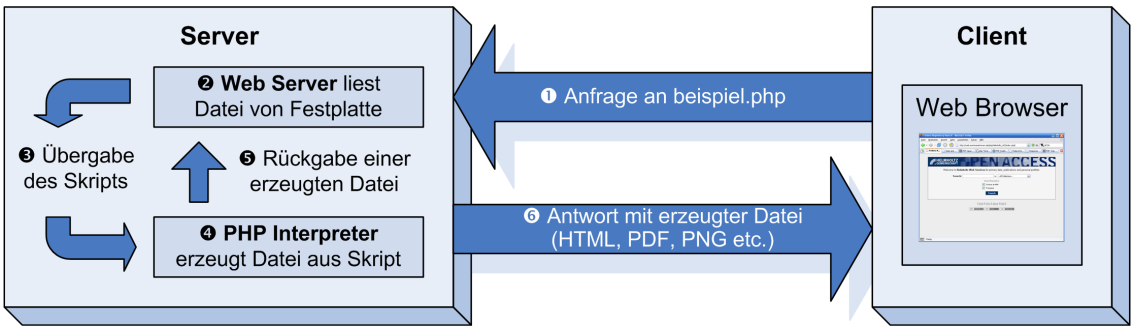


Abbildung 2.7: PHP: Funktionsweise (vgl. [Wik05b])

Beschreibung des Ablaufs:

[35] vgl. [Jen04]
[36] vgl. [PHP05, Kapitel 1]

1. Zunächst startet ein Client (Web Browser) eine Anfrage für eine PHP-Datei. Dies geschieht über die Eingabe einer URL oder durch anklicken eines Links. Die Anfrage wird über HTTP an den entsprechenden Server weitergeleitet.
2. Der Server (Web Server) nimmt die Anfrage entgegen und liest die PHP-Datei von seiner Festplatte.
3. Anschließend übergibt der Webserver die PHP-Datei an den PHP-Interpreter.
4. Der PHP-Interpreter arbeitet das PHP-Skript ab.
5. Anschließend gibt der Interpreter das Ergebnis seiner Arbeit (meist in Form einer HTML-Datei, aber auch Bilder oder PDF-Dokumente sind möglich) an den Web Server zurück.
6. Der Web Server liefert anschließend die Daten an den Client (hier an den Web Browser) zurück.

2.5.4.2 Integration in (X)HTML

PHP-Quelltext steht zwischen speziellen Anfangs- (`<?php`) und Schlusstags (`?>`). Durch diese ist es möglich PHP-Anweisungen an beliebiger Stelle eines HTML-Dokumentes auszuführen. Folgendes beispielhafte Listing erzeugt eine HTML-Datei, die auf dem Bildschirm „Hallo Welt“ ausgibt:

```
1 <html>
2  <head>
3   <title>Beispiel</title>
4  </head>
5  <body>
6   <p><?php echo "Hallo Welt!"; ?></p>
7  </body>
8 </html>
```

Quelltext 2.4: Programmbeispiel PHP

Da in der aktuellen PHP Version (5.0) ein Modul zur Verarbeitung von Web Services mittels SOAP integriert ist (PHPSOAP) und verstärkt XML-Unterstützung in PHP implementiert wurde (welche allerdings teilweise noch sehr experimentell ist), soll zur Programmierung der Anwendung PHP verwendet werden.

Kapitel 3

Überlegungen zur Leistungsoptimierung

3.1 Konzepte zur Abfrage von Daten

Grundsätzlich lässt sich sagen, dass Metadaten und die Ideologie die dahinter steht dafür zuständig ist, Objekte (physischer, wie auch elektronischer Art) in einer sachlichen Zusammenfassung darzustellen, um sie zur Vergleichbarkeit untereinander aufbereiten zu können, sie zu Gruppierungen zuzuordnen, um im Endeffekt eine Suche auf Informationen durchführen zu können, ohne den Inhalt eines Objektes direkt kennen zu müssen. Dieses schafft ungemein Vorteile in Bezug auf Effizienz und Speichervolumen, allerdings nur, wenn die Qualität der Metadaten stimmt – ein Katalog mit unvergleichbaren Inhalten ist unnütz und dient lediglich dazu, den Speicher als Abstellraum zu missbrauchen.

Informationen effizient wiederzufinden muss ein zu erstrebendes Ziel bei der Speicherung von Daten jedweder Art sein. Eine fortwährend anwachsende Informationsvielfalt, wie sie derzeit beispielsweise das Internet charakterisiert oder wie es die Forderung nach Open Access an deutschen Forschungseinrichtungen aufwirft, stellt dazu allerdings immense Anforderungen an Software, um adäquate Lösungen für Open Access realisierbar zu machen.

Da beide im Folgenden erläuterte Ansätze auf die Verwendung von Metadaten zurückgreifen, ja sogar beide das selbe Metadatenschema verwenden, ist grundlegend eine Ebene erreicht, auf der beide Konzepte technologisch, sowie ökonomisch gegenübergestellt werden können. Welcher der beiden Konzepte nun im Leistungsvergleich für die Zwecke von Open Access im Rahmen von Institutionellen Repositories für Publikationen unter dem Strich besser abschneidet, soll im Anschluss diskutiert werden.

Als Arbeitsumfeld zum Vergleich der Technologien dient das Repository-System *Fedora*. Es bietet jeweils Schnittstellen für OAI-PMH und Web Services, über die

Operationen auf den Datenbestand erfolgen können. Beide Schnittstellen sind so implementiert, dass sie ähnliche, beinahe identische Funktionen bereitstellen, was den Vergleich untereinander vereinfacht.

Im Folgenden soll mit einer Erläuterung des OAI-API's begonnen werden. Es ist über das HTTP-Protokoll aufrufbar und mittels OAI-PMH (OAI-Protocol for Metadata Harvesting) implementiert.

3.1.1 OAI-PMH

OAI-PMH scheint probate Möglichkeiten zu bieten, breiten Zugang zu verteilten Archiven zu schaffen. Die Firma *Google Inc.*, Betreiber der bekannten gleichnamigen Suchmaschine, nutzt bereits in einem ihrer Beta-Tests[1] das Harvesting von Archiven mittels OAI-PMH.

Propagiert wird, dass OAI-PMH einfach zu implementieren ist[2]. Ob eine OAI-PMH-Implementierung als geeignete Lösung zur Realisierung von Archiven für Institutionen mit hohem Output an Publikationen angesehen kann, und ob diese Implementierung für eine Langzeitarchivierung ausreicht, soll im Anschluss thematisiert werden.

3.1.1.1 Technik

OAI-PMH ist ein XML-basiertes Protokoll, das zur Abfrage und zum Übertragen von Metadaten dient. Dabei verwendet es das Dublin-Core Metadatenschema[3]. Es kann Metadaten von so genannten *Data-Provider* sammeln, kann diese aufbereiten und sorgt dafür, dass so genannte *Service-Provider* die gesammelten Daten mit Funktionen zur Suche versehen können. Ein Service Provider ist beispielsweise durch eine Weboberfläche mit Suchformular realisiert, durch welches auf die gesammelten Daten eines Harvesting-Prozesses zugegriffen werden kann.

3.1.1.2 Architektur

Die Idee die dahinter steht lässt sich mit der folgenden Grafik veranschaulichen.

[1] *Google Scholar*: Weltweite virtuelle Bibliothek wissenschaftlicher Arbeiten, noch im Beta-Stadium - Abrufbar im Internet unter `http://scholar.google.com`

[2] vgl. [Car03, Section 2 "'History and development of OAI-PMH"']

[3] s. Abschnitt 2.1.1, Seite 7

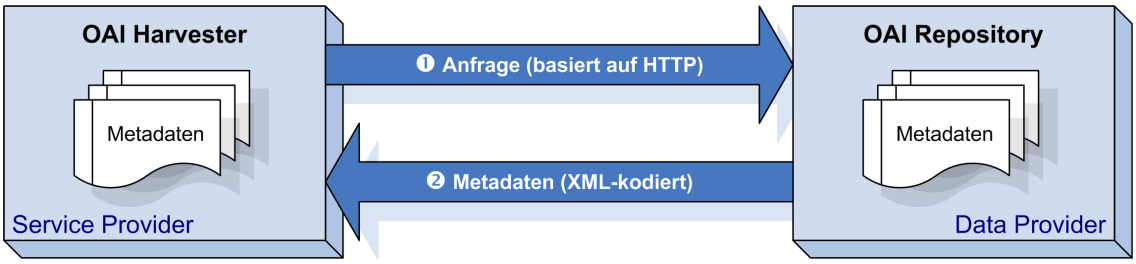


Abbildung 3.1: Grundlegende Funktionsweise von OAI-PMH (vgl. [Car03])

Ein Service-Provider startet einen Harvesting-Prozess um Metadaten eines Data-Providers abzufragen, welcher eine Schnittstelle für OAI-PMH bereitstellt. Der Service-Provider speichert die Ergebnisse der Anfrage in einer eigenen Datenbank, die nach dem abgeschlossenen Harvesting-Prozess für eine weitere Verwendung mit geeigneten Suchalgorithmen aufbereitet und nutzbar gemacht wird. Die Anfrage auf den Data-Provider geschieht dabei über HTTP – die generierte Antwort des Data-Provider ist XML-kodiert. Man spricht von Service- beziehungsweise Data-Provider auch gleichbedeutend von *Harvester*[4] und *Repository*[5].

Service-Provider bieten darüberhinaus die Eigenschaft, Daten von mehr als einem Data-Provider abzufragen und in einem eigenen, großen Archiv zu speichern. Die einzelnen Transaktionen werden dabei aufeinander folgend durchgeführt:

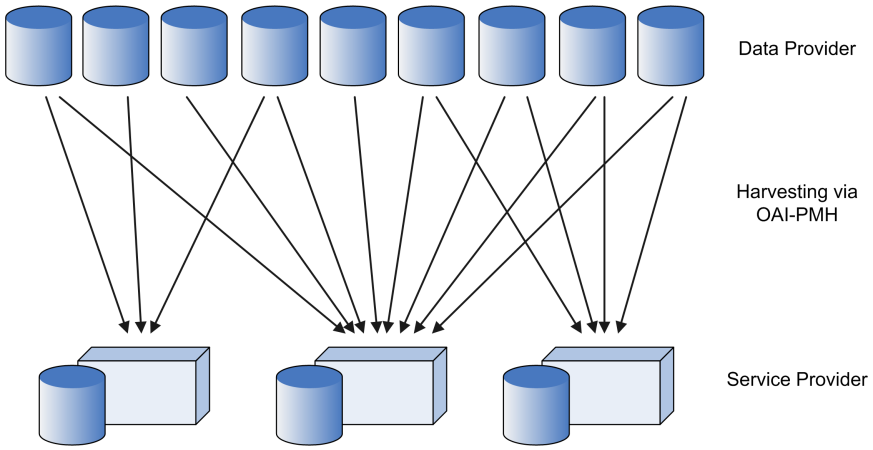


Abbildung 3.2: Harvesting-Prozesse mehrerer Service- und Data-Provider (vgl. [Car03])

Oft bieten Service-Provider die Möglichkeit ihren Datenbestand als Data-Provider wieder zur Verfügung zu stellen. Man spricht dann von so genannten *Aggregatoren* (engl. *Aggregators*). Abbildung 3.3 zeigt eine solche „Sammelstelle“, welche selbst

[4] dt.: „Erntemaschine“
[5] s. dazu auch Abschnitt 2.4, Seite 20

Daten abruft und gleichzeitig zum Abruf für weitere Service-Provider bereitstellt – ein Aggregator ist demnach Service- *und* Data-Provider zugleich.

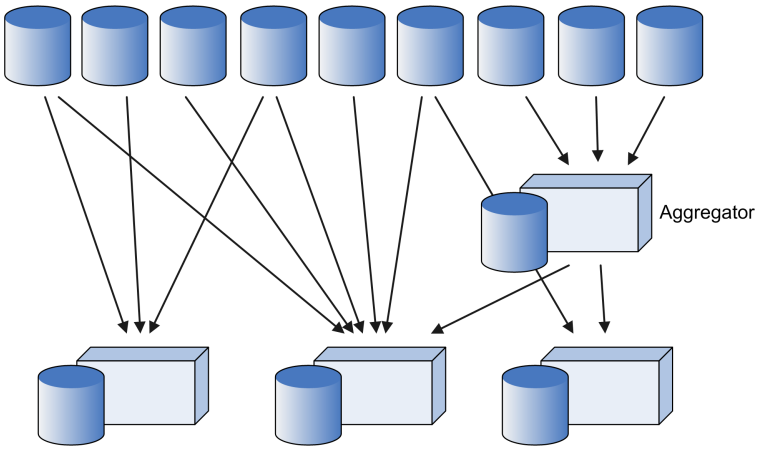


Abbildung 3.3: Zwischenschalten eines Aggregatoren (vgl. [Car03])

Alle Funktionen, die durch OAI-PMH zur Verfügung stehen, sind in der folgenden Abbildung im Überblick zusammenfassend dargestellt. Die rechts stehenden Archive besitzen OAI-PMH-Schnittstellen und fungieren somit als Data-Provider. Der Service-Provider „erntet“ die bereitgestellten Metadaten der angeschlossenen Archive. Diese stellt er weiteren Harvester bereit oder gibt sie beispielsweise einem Endnutzer über ein Web-Interface aus. Der Handlungsraum, in dem OAI-PMH agiert, ist im Folgenden mit gestrichelter Einfassung markiert:

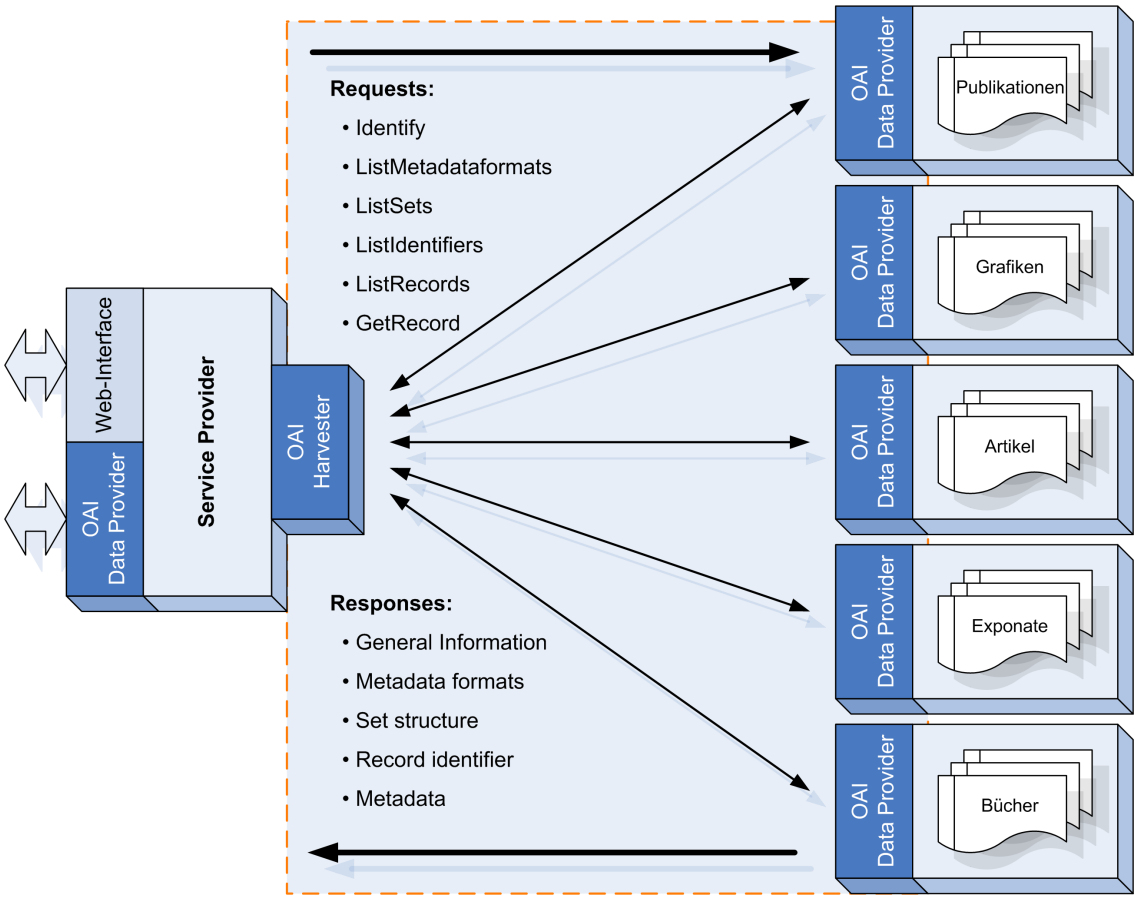


Abbildung 3.4: OAI-PMH: Überblick der Struktur und Funktionen (vgl. [Car03])

3.1.1.3 Funktionen

Wie bereits in der obigen Grafik dargestellt, stellt OAI-PMH eine Reihe von Mechanismen (so genannte *Verbs*[6]) zur Verfügung, durch welche auf Datensätze eines Data-Providers zugegriffen werden kann. Diese sind im Data-Provider implementiert und von einem Service-Provider über HTTP abrufbar.

- *Identify*
  Gibt Informationen über das Archiv zurück. Unter Anderem werden Name, BaseURL[7] verwendete Protokollversion und weitere grundlegende Informationen zur Identifikation des Archivs zurückgeliefert.

[6] vgl. [OAI04]

[7] BaseURL: Endpunkt eines Open Archives - Über einen Endpunkt identifiziert sich ein Repository, gleichzeitig bildet er die Schnittstelle für OAI-PMH-Anfragen

- *ListMetadataFormats*
  Liefert die verwendeten Metadaten-Schemata des Archivs zurück. Beispielsweise `oai_dc` für das Dublin-Core Metadatenschema.

- *ListSets*
  In einem Archiv können Datenbestände individuell verschiedenen *Sets* zugeordnet werden. Dies ist durchaus gängige Praxis zur Gruppierung von Inhalten. Wird *ListSets* ausgeführt, werden die Namen der verwendeten *Sets* zurückgegeben. Später kann so *selektives Harvesting*[8] auf das Archiv angewendet werden.

- *ListRecords*
  Mit dieser Funktion ist es möglich, Datensätze eines Archives zu „ernten“. Mit Angabe von optionalen Parametern ist es dabei möglich Teilmengen des Archivs herunterzuladen.

- *ListIdentifiers*
  Dieses *Verb* ist eine abgewandelte Form des *ListRecords*-Requests: Er gibt nur die Kopfdaten eines Datensatzes zurück.

- *GetRecord*
  Liefert die Daten eines bestimmten Datensatz nach Angabe einer Identifikation zurück.

OAI-PMH-Request Um die Funktionen, die durch *Verbs* bereitgestellt werden, nutzen zu können, müssen spezielle Anfragen formuliert und an ein OAI-Repository gesendet werden. Folgender beispielhafter HTTP-Request führt die *GetRecord*-Operation auf dem Fedora-Repository mit der BaseURL `http://web.awi-bremerhaven.de/fedora/oai` aus. Gesucht wird ein Element mit der Identifikation `Ola2001a` und dem Namespace `oai:awi-bremerhaven.de:awi`. Dabei sollen die Inhalte des Dublin-Core Metadataschemas (`oai_dc`) zurückgegeben werden (aus Darstellungsgründen wurde die URL umgebrochen):

```
http://web.awi-bremerhaven.de/fedora/oai?verb=GetRecord&metadataPrefix=oai_dc&identifier=oai:awi-
    bremerhaven.de:awi:Ola2001a
```

Quelltext 3.1: OAI-PMH Request

OAI-PMH-Response Die Antwort auf den vorangegangenen *GetRecord*-Request ist vom Typ *Metadata* und wie folgt aufgebaut:

[8] Abfrage von Teilmengen eines Archivs; vgl. [OAI04, Section 2.7.2 "‘Selective Harvesting and Sets"’]

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/
    OAI/2.0/OAI-PMH.xsd">
 <responseDate>2005-08-01T18:22:04Z</responseDate>
 <request verb="GetRecord" identifier="oai:awi-bremerhaven.de:awi:Ola2001a" metadataPrefix="oai_dc">
     http://web.awi-bremerhaven.de/fedora/oai</request>
 <GetRecord>
  <record>
   <header>
    <identifier>oai:awi-bremerhaven.de:awi:Ola2001a</identifier>
    <datestamp>2005-07-19T13:51:27Z</datestamp>
    <setSpec>objects</setSpec>
   </header>
   <metadata>
    <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/
        dc/elements/1.1/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
        "http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd
        ">
     <dc:title>Age estimates of isochronous reflection horizons by combining ice core, survey, and
         synthetic radar data.</dc:title>
     <dc:creator>Eisen, O.</dc:creator>
     <dc:creator>Nixdorf, U.</dc:creator>
     <dc:creator>Wilhelms, F.</dc:creator>
     <dc:creator>Miller, H.</dc:creator>
     <dc:subject>Geo System; Structure and Dynamics of the Lithosphere and Polar Ice Sheets; POL-
         MARCOPOLI</dc:subject>
     <dc:subject>EARTH SCIENCE &> Cryosphere &> Snow/Ice;</dc:subject>
     <dc:description>doi:10.1029/2003JB002858</dc:description>
     <dc:description>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=
         Ola2001a</dc:description>
     <dc:publisher>Journal of Geophysical Research-Solid Earth, 109, B04106, 2004</dc:publisher>
     <dc:date>2004-01-01</dc:date>
     <dc:type>text, article</dc:type>
     <dc:format>application/pdf</dc:format>
     <dc:identifier>doi:10.1029/2003JB002858</dc:identifier>
     <dc:identifier>Ola2001a</dc:identifier>
     <dc:identifier>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=
         Ola2001a</dc:identifier>
     <dc:identifier>awi-n11260</dc:identifier>
     <dc:identifier>awi:Ola2001a</dc:identifier>
     <dc:rights>uri:http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode</dc:rights>
    </oai_dc:dc>
   </metadata>
  </record>
 </GetRecord>
</OAI-PMH>
```

Quelltext 3.2: Beispiel OAI-PMH-Response

Die Antwort besteht aus mehreren Teilen und lässt sich wie folgt entschlüsseln:

1. Das erste Tag der XML-Datei ist die XML-Deklaration. Die XML-Version ist immer `1.0`, der verwendete Zeichensatz ist immer `UTF-8`:
   `<?xml version="1.0"encoding="UTF-8"?>`

2. Der weitere Inhalt ist in das Wurzel-Element `OAI-PMH` eingeschlossen. Es besteht immer aus drei Attributen, welche die Namespaces und Schemata für den gesamten Inhalt des Responses enthält.

- `xmlns` – muss den Namespace (URI) des gegenwärtigen OAI-PMH enthalten: `http://www.openarchives.org/OAI/2.0/`.
- `xmlns:xsi` – enthält den Namespace (URI) des XML-Schemas: `http://www.w3.org/2001/XMLSchema-instance`.
- `xsi:schemaLocation` – ist ein zusammengesetzte Attribut: Der erste Teil ist der Namespace (URI) (wie in der XML-Spezifikation definiert) des OAI-PMH (`http://www.openarchives.org/OAI/2.0/`), der zweite Teil ist die URL des XML-Schemas, gegen welches der Response validiert werden muss (`http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd`).

3. Bei allen OAI-PMH-Responses sind die ersten zwei Kind-Elemente Folgende:
   - `responseDate` – ein Datum in UTC[9]-Format, welche die Zeit angibt, wann der Response gesendet wurde.
   - `request` – zeigt die Methode an, die im Request angefordert wurde und die vorliegende Antwort generierte. Nach den folgenden Regeln ist das `request`-Element aufgebaut:
     - Inhalt des Elements ist die BaseURL des Repository, an die der Request gestellt wurde, bzw. von dem der Response ausgeht. Hier: `http://web.awi-bremerhaven.de/fedora/oai`
     - Die Attribute müssen Parameter der Funktionsaufrufe des jeweiligen Verbs sein. Im obigen Beispiel wurde das getRecord-Verb aufgerufen, welches die Parameter `identifier` und `metadataPrefix` erfordert. Eine Parameter-Referenz aller Funktionen ist in [OAI04] nachzuschlagen.
     - Falls ein Fehler auftritt (`badVerb` oder `badArgument`), darf das `request`-Element lediglich die BaseURL zum Inhalt haben. Es darf *keine* Attribute enthalten.

4. Drittes Kind-Element ist entweder:
   - ein `error`-Element, welche im Fall eines Fehlers Details an die aufrufende Stelle zurückliefert

     oder, wie in diesem Beispiel
   - ein Element mit dem selben Namen des Verbs, welches den OAI-PMH-Request auslöste. Im obigen Beispiel ist dies das `GetRecord`-Element, welches die Metadaten eines Objektes enthält. Es besteht aus `header` und `metadata`-Tag. Ersteres enthält Informationen zum angeforderten Objekt (Identifizierung, Datum und Typ), das Zweite enthält die zugehörigen Metadaten. Hier wurde das Metadatenschema `oai_dc` (Dublin Core) angefordert, welches durch die XML-Tags mit dem Präfix „`dc:`“ dargestellt werden.

[9] Coordinated Universal Time: Koordinierte Weltzeit, Referenzzeit, von der die Zeiten in den verschiedenen Zeitzonen der Erde abgeleitet werden, ausgehend vom Null-Meridian, vgl. [WW98]

3.1.2 Web Services

Mittlerweile bieten Repository-Systeme allerdings auch alternative Methoden zur Verfügung, die Zugriffe auf Datenbestände erlauben[10]. Mittels Web Services wird dazu, parallel zu den klassischen OAI-PMH-Implementierungen, eine weitere Schnittstelle integriert – das Simple Object Access Protocol (SOAP) übernimmt dabei den Datentransport.

3.1.2.1 Architektur zur Abfrage und Übertragung von Daten

Der Ansatz zur Datenübertragung ist ähnlich dem des OAI-PMH, allerdings ist durch Web Services nicht festgelegt, welche Art von Daten übertragen werden soll – dies kann anwendungsspezifisch festgelegt werden. Web Services stellen also nur die Technik zur Verfügung, nicht aber vordefinierte Funktionen. Folgendes Schaubild zeigt den Ablauf einer grundlegenden Datenübertragung bei der Verwendung von Web Services in Verbindung mit SOAP. Transportiert werden in diesem Beispiel die Metadaten eines Objektes aus einem Repository:

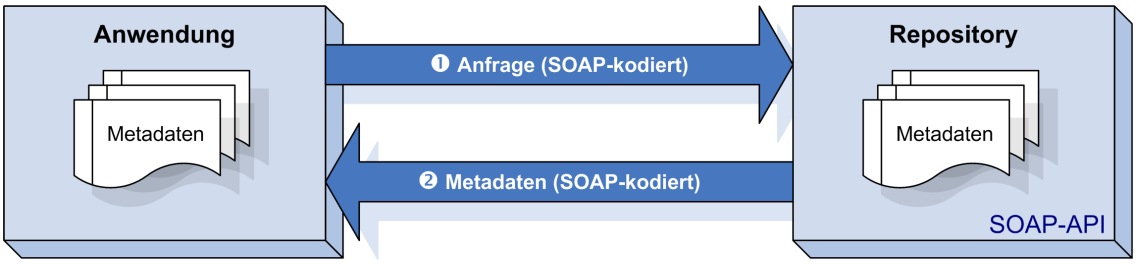


Abbildung 3.5: Grundlegende Funktionsweise der Kommunikation von Web Services

Im Gegensatz zu OAI-PMH können mit Web Services Anfragen an mehrere Repositories gleichzeitig durchgeführt werden. Ein SOAP-Client formuliert dazu einen SOAP-Request, welcher Angaben über die angeforderten Daten enthält (ein Suchwort beispielsweise) und sendet diesen an die Repositories. Nachdem die Anfragen auf Repository-Seite verarbeitet und ein (Such-)Ergebnis ermittelt wurde, wird jeweils ein SOAP-Response formuliert, der das Ergebnis an den Aufrufer zurücksendet. Dieser kann die Antworten aller angesprochenen Repositories sortiert in einer beliebigen Anwendung an die eigenen Bedürfnisse angepasst aufbereiten und ausgeben. Der Aufrufer kann dabei selbst entscheiden, ob er eine Datenbank zur Speicherung der Ergebnisse verwendet oder diese nur temporär im Zwischenspeicher behält. Letztere ist die gängige Vorgehensweise bei der Abfrage und Übertragung von Daten mit Web Services.

[10] vgl. [BOA04, Seite 17]

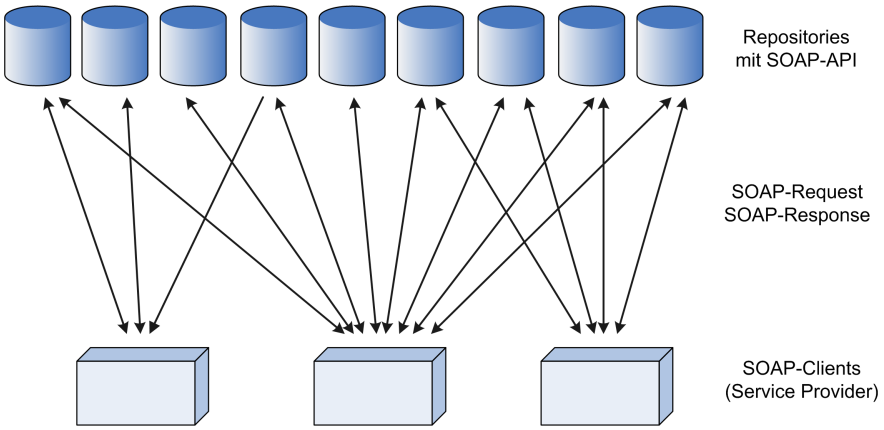


Abbildung 3.6: Funktionsweise von Abfragen bei Respositories mit SOAP-API

3.1.2.2 SOAP-API's

Client-Anwendungen können, wie mittels OAI-PMH, durch eine SOAP-API auf die Datenhaltung des Repository mit Web Services-Schnittstelle zugreifen. Je nach Repository sind die bereitgestellten Funktionen allerdings individuell implementiert, so dass bei der Entwicklung von geeigneten Clients die angebotenen Funktionalitäten individuell bedient werden müssen (Gründe: Funktionen nicht implementiert; unterschiedliche Funktionsnamen; differierende Parameter zum Aufruf einer Funktion; etc.). Allerdings heben sich SOAP-API's von Implementierungen mittels OAI-PMH mit einer entscheidenden Eigenschaft ab: Zusätzlich können Funktionen bereitgestellt werden, die *Änderungen* am Datenbestand durchführen können – es ist (je nach verwendetem Repository) möglich, Daten hinzuzufügen, sie zu verändern und sogar zu löschen. Das in dieser Arbeit verwendete Repository bietet als eines der Ersten jeweils solche SOAP-API's zum Lese- *und* Schreibzugriff an (Stand: August 2004)[11].

Um den Zugriff auf die Funktionen der SOAP-Implementierung zu regeln wurden in dem hier verwendeten Repository-System zwei verschiedene API's mit unterschiedlichen Zugriffsrechten und Authentifizierungsmaßnahmen in das System integriert. Zu diesen beiden API's gibt es zusätzlich jeweils funktionsreduzierte Versionen (LITE-API's). Da diese aber nur einfachste Methoden zum Datenzugriff anbieten, die im Gegensatz zu den API's mit vollem Funktionsumfang nicht für die Zwecke dieser Arbeit ausreichen, werden diese im Folgenden nicht erläutert.

Fedora-API-A Der Web Service der hinter Fedora's API-A[12] steht, ist zuständig für den Lesezugriff auf digitale Objekte. Es stellt öffentliche Methoden zur Verfü-

[11] vgl. [BOA04, Seite 11]
[12] *A* steht als Akronym für *Access* (dt.: *Zugriff*)

gung, um die Eigenschaften von Objekten anzuzeigen (bspw. die enthaltene Metadaten). Hauptaufgabe des API-A-Service ist die Abarbeitung eines Client-Requests zur Ausgabe von angeforderten Ansichten von Objektdaten. Dazu wertet das zugrunde liegende Repository die Verknüpfungseigenschaften der angeforderten digitalen Objekte aus und führt gegebenenfalls interne Modifikationen an der Objektrepräsentation mittels internen oder externen (selbst festlegbaren) Service-Funktionen durch, um schließlich eine gewünschte Ausgabe zu erzeugen und an einen Client zurückzuliefern.

Des Weiteren stellt dieses API Funktionen bereit, um individuelle Eigenschaften des Repository abzufragen. Im Folgenden sind die Methoden der Fedora-API-A aufgelistet, wie sie der integrierte Web Service zur Verfügung stellt.

- `describeRepository`
  Gibt beschreibende Informationen über das Repository ähnlich des OAI-Verbs *Identify* zurück. Unter anderem enthält der Response Name, Version, BaseURL, PID-Namespace und Beispiel URLs zum Stellen von Anfragen
- `findObjects`
  Übergabe von Suchparametern; Zurückliefern von Objekten, die den Kriterien entsprechen.
- `resumeFindObjects`
  Fortführung eines `findObjects`-Requests.
- `getObjectProfile`
  Detailinformationen über ein bestimmtes Objekt zu einem bestimmten Zeitpunkt werden angezeigt.
- `getObjectHistory`
  Zeigt die bereits durchgeführten Änderungen an einem Objekt an.
- `listDatastreams`
  Gibt alle Datastreams eines Objektes zu einem angegebenen Zeitpunkt zurück.
- `listMethods`
  Die mit Behaviour-Mechanismen implementierten Methoden werden angezeigt.
- `getDatastreamDissemination`
  Gibt eine bestimmte Transformation eines Objektes zurück.
- `getDissemination`
  Details über einen Mechanismus zur Transformation werden mit dieser Methode zurückgegeben.

Fedora-API-M Fedora's Management Web Service (API-M[13]) erweitert die bereits erläuterten Funktionen des API-A um Methoden zur Verwaltung des Datenbestandes. Sie stellt eine offene Administrations-Schnittstelle zur Verfügung, mit der es möglich ist digitale Objekte (und zugehörige Komponenten) zu erstellen, zu modifizieren und zu löschen. Dazu kommuniziert der Web Service mit den Kern-Funktionen des Repository-Systems, um bestehende Inhalte zu lesen oder neue Inhalte an die Repository-interne Datenhaltung zu übergeben.

Somit wird durch das API-M eine Reihe von lose gekoppelten Funktionen zur Administration zur Verfügung gestellt. Das heißt, ein Client, der auf Administrationsfunktionen zugreift, braucht die zugrunde liegenden Speicherformate, Medien oder Regeln des Speichermanagements nicht zu kennen.

Da der Fokus dieser Arbeit allerdings auf Funktionen des lesenden Datenzugriffs liegt, wird auf die Funktionen des API-M nicht weiter eingegangen. Diese können allerdings in [Fed02] nachgeschlagen werden.

3.1.2.3 Wichtige Methoden der Fedora-API-A

Genauer sollen die zwei folgenden Methoden untersucht werden, da sie in gleicher Weise aufgebaut sind, wie die Funktionen zur Abfrage von Metadaten durch OAI-PMH.

Methode: „findObjects“

Funktion	Suchfunktion, welche die angeforderten Objekt-Felder aller Objekte des Repository zurückliefert, die den angegebenen Kriterien entsprechen.
Parameter	`resultFields`: Ein Array eindeutiger Feldbezeichner, wie sie in Fedora verwendet werden. Diese Angabe legt fest, welche Feldinhalte zurückgeliefert werden sollen `maxResults`: Die maximale Anzahl der Ergebnisse, die pro Response geliefert werden sollen. Hier ist der maximale Wert durch die Fedora-Konfiguration eingeschränkt (vgl. Punkt 6.1.1.4, Seite 77) `query`: Suchbegriffe, bzw. Suchkonditionen
Rückgabe	Enthält ein Array, gefüllt mit Inhalten aus den angegebenen Objektfeldern und eventuell eine Session-ID, falls die Anzahl der Ergebnisse die Listengröße überschreitet.

Tabelle 3.1: Fedora-API-A: findObjects

[13] *M* steht als Akronym für *Management* (dt.: *Verwaltung*)

Methode: „resumeFindObjects“

Funktion	Liefert die nächste folgende Liste mit Ergebnissen zurück, falls die vorangegangene Ergebnisliste der Methode `findObjects` abgeschnitten wurde.
Parameter	`sessionToken`: eindeutige Session-ID, die aus dem vorangegangenen `findObjects`-Request übermittelt wurde
Rückgabe	Enthält ein Array, gefüllt mit einer Fortsetzung der Inhalte eines vorangegangenen `findObjects`-Requests und eventuell eine weitere Session-ID, falls die Anzahl der Ergebnisse die Listengröße überschreitet.

Tabelle 3.2: Fedora-API-A: resumeFindObjects

SOAP-Request (findObjects)

Eine Anfrage mittels `findObjects` wird durch den Web Service-Benutzer in folgender Form verfasst und an das Repository gesendet:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.
       w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <soapenv:Body>
4    <ns1:findObjects soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="
         http://www.fedora.info/definitions/1/0/api/">
5     <resultFields xsi:type="soapenc:Array" soapenc:arrayType="xsd:string[16]" xmlns:ns2="http://www.
          fedora.info/definitions/1/0/types/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
          ">
6      <item>pid</item>
7      <item>title</item>
8      <item>creator</item>
9      <item>subject</item>
10     <item>description</item>
11     <item>publisher</item>
12     <item>contributor</item>
13     <item>date</item>
14     <item>type</item>
15     <item>format</item>
16     <item>identifier</item>
17     <item>source</item>
18     <item>language</item>
19     <item>relation</item>
20     <item>coverage</item>
21     <item>rights</item>
22    </resultFields>
23    <maxResults xsi:type="xsd:nonNegativeInteger">100</maxResults>
24    <query href="#id0"/>
25   </ns1:findObjects>
26   <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
         encoding/" xsi:type="ns3:FieldSearchQuery" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
         encoding/" xmlns:ns3="http://www.fedora.info/definitions/1/0/types/">
27    <conditions xsi:type="ns3:Condition" xsi:nil="true"/>
28    <terms xsi:type="xsd:string">01a2001a</terms>
29   </multiRef>
30  </soapenv:Body>
31 </soapenv:Envelope>
```

Quelltext 3.3: findObjects-SOAP-Request

Nachdem das Repository die Anfrage verarbeitet hat, wird der SOAP-Request aus der `findObjects`-Anfrage generiert und an den Aufrufer zurückgeschickt.

SOAP-Response (findObjects)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.
       w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <soapenv:Body>
4    <ns1:findObjectsResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:ns1="http://www.fedora.info/definitions/1/0/api/">
5     <response href="#id0"/>
6    </ns1:findObjectsResponse>
7    <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
         encoding/" xsi:type="ns2:FieldSearchResult" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
         encoding/" xmlns:ns2="http://www.fedora.info/definitions/1/0/types/">
8     <listSession xsi:type="ns2:ListSession" xsi:nil="true"/>
9     <resultList xsi:type="soapenc:Array" soapenc:arrayType="ns2:ObjectFields[1]">
10     <item href="#id1"/>
11    </resultList>
12   </multiRef>
13   <multiRef id="id1" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
         encoding/" xsi:type="ns3:ObjectFields" xmlns:ns3="http://www.fedora.info/definitions/1/0/
         types/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
14    <pid xsi:type="xsd:string">awi:Ola2001a</pid>
15    <label xsi:type="xsd:string" xsi:nil="true"/>
16    <fType xsi:type="xsd:string" xsi:nil="true"/>
17    <cModel xsi:type="xsd:string" xsi:nil="true"/>
18    <state xsi:type="xsd:string" xsi:nil="true"/>
19    <ownerId xsi:type="xsd:string" xsi:nil="true"/>
20    <cDate xsi:type="xsd:string" xsi:nil="true"/>
21    <mDate xsi:type="xsd:string" xsi:nil="true"/>
22    <dcmDate xsi:type="xsd:string" xsi:nil="true"/>
23    <title xsi:type="xsd:string">Age estimates of isochronous reflection horizons by combining ice
          core, survey, and synthetic radar data.</title>
24    <creator xsi:type="xsd:string">Eisen, O.</creator>
25    <creator xsi:type="xsd:string">Nixdorf, U.</creator>
26    <creator xsi:type="xsd:string">Wilhelms, F.</creator>
27    <creator xsi:type="xsd:string">Miller, H.</creator>
28    <subject xsi:type="xsd:string">Geo System; Structure and Dynamics of the Lithosphere and Polar
          Ice Sheets; POL-MARCOPOLI</subject>
29    <subject xsi:type="xsd:string">EARTH SCIENCE &amp;gt; Cryosphere &amp;gt; Snow/Ice; </subject>
30    <description xsi:type="xsd:string">doi:10.1029/2003JB002858</description>
31    <description xsi:type="xsd:string">uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/
          abstract.php?uid=Ola2001a</description>
32    <publisher xsi:type="xsd:string">Journal of Geophysical Research-Solid Earth, 109, B04106, 2004</
          publisher>
33    <date xsi:type="xsd:string">2004-01-01</date>
34    <type xsi:type="xsd:string">text, article</type>
35    <format xsi:type="xsd:string">application/pdf</format>
36    <identifier xsi:type="xsd:string">doi:10.1029/2003JB002858</identifier>
37    <identifier xsi:type="xsd:string">Ola2001a</identifier>
38    <identifier xsi:type="xsd:string">uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/
          abstract.php?uid=Ola2001a</identifier>
39    <identifier xsi:type="xsd:string">awi-n11260</identifier>
40    <identifier xsi:type="xsd:string">awi:Ola2001a</identifier>
41    <rights xsi:type="xsd:string">uri:http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode</
          rights>
42   </multiRef>
43  </soapenv:Body>
44 </soapenv:Envelope>
```

Quelltext 3.4: findObjects-SOAP-Response

In diesem Beispiel wurde nach dem `identifier` des Objektes „`Ola2001a`“ gesucht (siehe `terms`-Element im Listing „findObjects-SOAP-Request“, Zeile 27), aus diesem

Grund besteht der SOAP-Response aus dem `resultList`-Element mit nur einem Kind-Element, da keine weiteren Objekte dem Suchkriterium entsprechen. Dieses Kind-Element enthält die Metadaten, die mit dem SOAP-Request angefordert wurden.

Würde die Anzahl der Treffer der Suchanfrage die im Repository-System festgelegte maximale Anzahl der Ergebnisse pro Request überschreiten, wird dem SOAP-Response eine Session-ID beigefügt, die einer `resumeFindObjects`-Anfrage übergeben werden kann. Mit `resumeFindObjects` werden somit weitere Ergebnisse angezeigt – solange, bis alle Ergebnisse abgerufen wurden. Der SOAP-Response der `resumeFindObjects`-Methode ist gleichermaßen aufgebaut, wie der des `findObjects`-Response.

3.2 Vergleich

Verglichen werden nun im Folgenden die Schnittstellen, welche Zugriffe auf Daten ermöglichen: Fedora-OAI-API (OAI-PMH) und Fedora-API-A (Web Services). Dieser Vergleich ist objektiv und soll nicht den Eindruck erwecken, die eine Technologie sei minderwertiger als die Andere, vielmehr soll die Untersuchung der Eigenschaften beider Architekturen ein Bild darüber erzeugen, welche Technologie zu welchen Anwendungszwecken eingesetzt werden sollte.

3.2.1 Gegenüberstellung der Konzepte

Beide Architekturen sind XML-basierend, dieses wurde bereits in den vorangegangenen Darstellungen der Architekturen grundlegend erläutert. Beide Konzepte erlauben es lesende Zugriffe auf einen Datenbestand durchzuführen. Dabei unterscheiden sich beide Schnittstellen allerdings so immens, sodass der folgende tabellarische Vergleich Aufschluss darüber geben soll, wo Stärken und wo Schwächen der jeweiligen Technologie liegen.

Im Folgenden werden die Begriffe „Server-Anwendung“ und „Client-Anwendung“ genannt. Mit „Server-Anwendungen“ sind im Folgenden die Systeme betitelt, welche als Data-Provider im Sinne von OAI-PMH eingesetzt werden, beziehungsweise die Rolle des Repository für die Betrachtungsweise bei Web Services einnehmen. Der Begriff „Client-Anwendung“ definiert die Systeme, die einen Zugriff auf die Server-Anwendungen durchführen um Daten abzurufen:

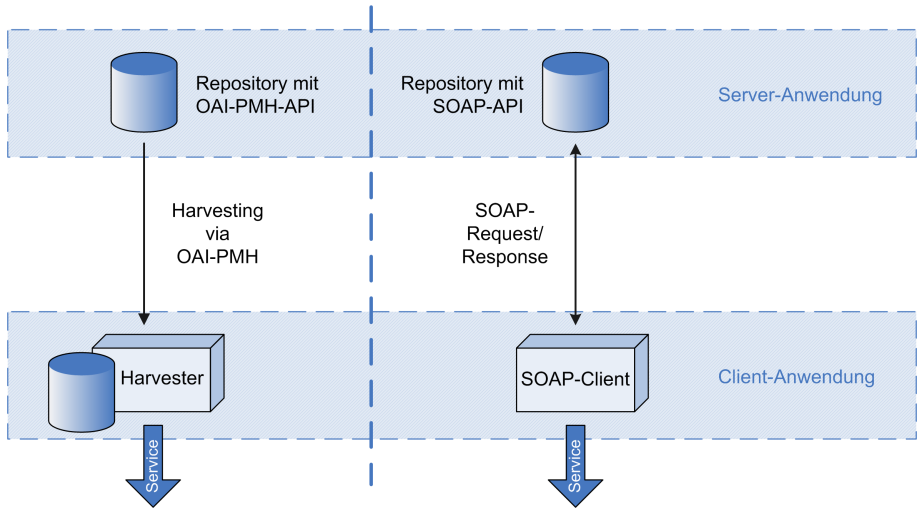


Abbildung 3.7: Gegenüberstellung der Konzepte

3.2.1.1 Grundvoraussetzungen und Implementierung

Wichtig zu wissen ist, dass beide Architekturen verschiedene Anforderungen an Server- bzw. Client-Anwendungen stellen:

	OAI-PMH	**Web Service** (*via Fedora*)
Anforderungen	**Server-Anwendung:** Web Server; Datenbank **Client-Anwendung:** Web Server; für große Datenbestände geeignete Datenbank, ggf. Indexdienst	**Server-Anwendung:** Web Server; Datenbank **Client-Anwendung:** Individuelle Anforderungen, da in beliebige (multithreading-fähige) Anwendung integrierbar
Aufwand der Implementierung	**Server-/Client-Anwendung:** Gering, es existieren frei verfügbare Werkzeuge zur Implementierung.	**Server-Anwendung:** Hoch, es existieren erst wenige verwendbare Repository-Systeme mit Web Service-basierten Schnittstellen; Viel Einarbeitungszeit erforderlich; Bei Fedora: Java-Programmierer notwendig. **Client-Anwendung:** Hoch, da Anwendungen oder Schnittstellen für Anwendungen für den Zugriff auf die Datenhaltung eines Repository individuell erstellt werden müssen.

Tabelle 3.3: OAI-PMH versus Web Services: Anforderungen und Aufwand der Implementierung

3.2.1.2 Interoperabilität

Die Interoperabilität ist wichtig zur Gewährleistung des breiten Einsatzes einer Architektur. Ist eine Technologie nicht standardisiert, wird die Akzeptanz und Nutzung derer auf geringem Niveau bleiben, da individuelle Anwendungen mit unterschiedlichen, inkompatiblen Schnittstellen entworfen werden können. Es entstehen dann Insellösungen, die schnell teuer werden können[14].

	OAI-PMH	**Web Service** (*via Fedora*)
Grad der Standardisierung	Alle Funktionen zum Zugriff auf Daten und der Übertragungsmechanismus sind durch die OAI standardisiert. Dies gewährleistet hohe Interoperabilität.	Nur Technologie ist standardisiert, Funktionsintegration zur Abfrage von Daten eines Repository ist anwendungsspezifisch. Interoperabilität für die Funktionsintegration ist nicht gegeben, für den Transport der Daten aber schon.

Tabelle 3.4: OAI-PMH versus Web Services: Standardisierung

3.2.1.3 Performance

	OAI-PMH	**Web Service** (*via Fedora*)
Suchen von Datensätzen	Wird auf lokalem Datenbestand der Client-Anwendung durchgeführt; Ist also von den *eigenen Suchalgorithmen* abhängig;	Wird auf den Repositories selbst durchgeführt, Ergebnisse werden an die Client-Anwendung zurückgeliefert; Hängt von der Implementierung der *Suchalgorithmen der einzelnen Repositories* (Server-Anwendung) ab.
Art der Datenübertragung	**Request:** HTTP-POST; **Response:** XML kodiert	**Request/-Response**: jeweils XML-kodiert
Volumen der Datenübertragung	**Ein Datensatz:** $\approx 2,4$ kB (vgl. Listings 3.1 und 3.2); **100 Datensätze:** $\approx$ 18 kB	**Ein Datensatz:** $\approx 4,7$ kB (vgl. Listings 3.3 und 3.4); **100 Datensätze:** $\approx$ 179 kB

Tabelle 3.5: OAI-PMH versus Web Services: Suche und Art/Volumen der Datenübertragung

Bei OAI-PMH sind große Archive nur mit hohem Zeitaufwand durchsuchbar: Abhilfe können effektive (evtl. teure) Indexdienste schaffen.

Im Gegensatz dazu können beim Einsatz der Web Service-Technologie gleichzeitige Anfragen an mehrere Repositories gesendet werden. So kann Zeit durch die gezielt parallele Abarbeitung der Abfrage durch die Repositories selbst eingespart werden. Ein Client muss also *selbst keine* rechenintensiven Suchalgorithmen durchlaufen, er gibt lediglich die *zurückgelieferten Ergebnisse* aus einem Repository direkt aus.

[14] vgl. [Don01]

3.2.1.4 Kostenabschätzung

Da beide Architekturen verschiedene Anforderungen an das System stellen, unterscheiden sie sich auch in der Kostenfrage.

	OAI-PMH	**Web Service** (*via Fedora*)
Kosten der Implementierung	**Server-Anwendung:** Kosten entstehen durch Web Server und Datenhaltung; Höhe der Kosten je nach Art der Datenhaltung allerdings verschieden, da eine Schnittstelle für OAI-PMH bei bereits bestehende Systemen geschaffen werden muss. **Client-Anwendung:** Datenbank und Web Server müssen zur Verfügung stehen.	**Server-Anwendung:** Eher geringere Kosten: Das Fedora-Repository bringt beispielsweise Web Server und Datenhaltung im Lieferumfang mit. Allerdings kann die Datenhaltung auch durch externe Systeme übernommen werden. **Client-Anwendung:** Eher geringere Kosten: Kosten entstehen durch eine individuelle (evtl. bereits bestehende) Anwendung; Es ist keine Datenbank notwendig.

Tabelle 3.6: OAI-PMH versus Web Services: Implementierungskosten

Die Anschaffungskosten bewegen sich in einem Rahmen, der wenig ins Gesamtgewicht fällt. Die Anschaffungskosten für Software sind vergleichsweise vernachlässigbar (durch Verwendung von Open Source Software) im Vergleich zu den Kosten, die durch die Implementierung und Systemintegration beider Systeme durch eingesetztes Personal entstehen. Hier entstehen je nach bestehenden Systemen allerdings verschieden große Aufwände, so dass nicht verallgemeinert werden kann, wie viel die eine oder andere Integration kosten wird. In der obigen Tabelle sind also lediglich grobe Abschätzungen für die Anschaffungskosten enthalten.

3.2.2 Ergebnisse der Gegenüberstellung beider Konzepte

Grundsätzlich unterscheiden sich die beiden untersuchten Konzepte zur Abfrage von (Meta)-Daten durch ihre verschiedenen Ansätze und Architekturen. Während OAI-PMH eine Plattform für Archive in monolithischem Stil anbietet – ein großes Archiv, in dem alle Daten gesammelt werden – bieten Architekturen die auf Web Services basieren einen Service-orientierten Ansatz. Das bedeutet, Daten werden erst dann abgerufen, wenn sich auch wirklich benötigt werden (Stichwort SOA[15]). Es gibt keine eigene Datenhaltung auf Seite der Client-Anwendung, der Zugriff erfolgt direkt und gleichzeitig auf die beim Daten-Anbieter hinterlegten Suchalgorithmen.

3.2.2.1 Rolle des Service-Provider

Dies bedeutet, ein Repository muss Suchalgorithmen bereitstellen, um Daten für den Service-orientierten Ansatz zur Verfügung stellen zu können. Da ein Repository allerdings Daten beinhaltet, die durch die Implementierung desselben in einer Institution oder einer Firma bereits mit Suchfunktionen versehen sind (z.B. zur internen Recherche), können diese ebenfalls für die Web Service-Implementierung eingesetzt werden. Suchalgorithmen müssen also (zumeist) nicht neu integriert werden, da sie schon indirekt zur Verfügung stehen. Sie müssen lediglich durch das Schaffen einer Schnittstelle für SOAP nutzbar gemacht werden – der Aufwand dazu ist nicht sonderlich hoch.

Die Rolle des „Service-Provider“ (im Sinne von OAI-PMH) ist beim Einsatz von Web Services verschoben. Jedes einzelne Repository tritt als Data- und Service-Provider gleichzeitig auf (sog. „Aggregator“). Der Web Service-orientierte Ansatz nutzt dabei die positiven Eigenschaften dieser entstandenen Service-orientierten Architektur. Der Vorteil entsteht dadurch, dass parallele Anfragen auf mehrere Data-Provider gleichzeitig möglich sind: Bei der Anfrage an mehr als ein Repository gibt es im Gegensatz zu der OAI-PMH-Implementierung einen enormen Zeitvorteil bei der Abarbeitung von Requests.

3.2.2.2 Aktualität der Daten

Weiterer Vorteil einer Web Services-Implementierung ist die fortwährende Aktualität der Daten. Web Services rufen immer die aktuellen Zustände der Objekte eines Repository ab, während OAI-Service-Provider immer einen „veralteten“ Datenbestand offerieren. Dieser ist immer so alt, wie der letzte Harvesting-Prozess.

Der Grund dafür ist, dass ein Harvester, je nach Größe des Datenvolumens eines Repository, nur in bestimmten Zeitabständen das Harvesting zur Auffrischung des Datenbestandes durchführen kann. Je größer das Datenvolumen ist, desto länger ist ein Harvester damit beschäftigt, seinen Datenbestand mit dem des Repository

[15] SOA: „Service-oriented Architecture“

abzugleichen. Dies hat zur Folge, dass gerade bei größeren Repositories ein großes Zeitintervall zwischen den Harvesting-Prozessen entsteht und somit das Alter der offerierten Daten linear mit der Datenmenge ansteigt.

Bei Datenbeständen von der Größenordnung um zehntausend Objekte fällt das nicht so immens ins Gewicht, da es im Gegensatz zu größeren Datenmengen keinen großen Zeitaufwand benötigt. Sind aber erst einmal größere Dimensionen erreicht (eine Million Objekte bspw.), wird sich ein Archiv welches lediglich OAI-PMH-Schnittstellen bereitstellt nicht besonders zum Harvesting eignen[16]. Spätestens dann müssen andere Architekturen eingesetzt werden, die einen alternativen Ansatz zum Datenzugriff bieten.

3.2.2.3 Traffic

OAI-PMH glänzt im Vergleich mit geringem Datenaufkommen bei der Übertragung von einzelnen Objekten eines Repository. Dazu muss allerdings berücksichtigt werden dass durch einen Harvesting-Prozess hoher Traffic im Gesamten entsteht: Ein auf einem Fedora-Repository durchgeführter Harvesting-Prozess benötigt initial 6 Minuten und 14 Sekunden mit einem Datenvolumen von $\approx 26,25$ MB um die in dem hier berücksichtigtem Test-Szenario enthaltenen 14.150 Objekte des Fedora Repository in die eigene Datenhaltung zu überführen[17]. Ein Update durch diesen (ohne jegliche Änderungen am Datenbestand auf Repository-Seite) dauerte 5 Minuten und 54 Sekunden mit einem verbundenem (unnötigem[18]) Datenvolumen von ebenfalls $\approx 26,25$ MB. (Wenn es Änderungen gegeben hätte, bliebe das Volumen und die Abarbeitungszeit entsprechend gleich – Es sei denn das Harvesting wird selektiv durchgeführt, dann werden allerdings auch nur Teilbereiche erneuert.)

OAI-Data-Provider können aufgrund dessen schnell überlastet sein, da Service-Provider selbst entscheiden können, wann und wie oft ein Harvesting-Prozess durchgeführt werden soll. Die Abarbeitung des Harvesting bleibt dabei am Data-Provider hängen – er muss jeden Service-Provider bedienen. Ein Data-Provider ist dabei zumeist schon durch *einen* Harvesting-Prozess für eine bestimmte Zeit (in dem hier berücksichtigen Szenario: rund 6 Minuten) voll damit ausgelastet Objekte zu suchen und auszuliefern, ein Zweiter verlangsamt die Abarbeitung um das Doppelte. Erschwerend kommt hinzu, wenn der Data-Provider populär ist, dass durchgehend gleichzeitige Harvesting-Prozesse die Abarbeitungszeit ins Unermessliche steigen

[16] Ein Harvester welcher das WDC-MARE-Repository „Pangaea“ abfragt, welches derzeit ca. 240.000 Objekte enthält, ist mit dem Harvesting rund 8 Stunden beschäftigt. Dies ergeben Erfahrungsberichte aus dem am AWI implementierten PKPHarvester.

[17] Als Harvester wurde im Test der „PKPHarvester“ mit lokaler MySQL-Datenbank auf einer Workstation eingesetzt (Systemeigenschaften: Intel Pentium 4, 2,8 GHz; 1 GB RAM; GigaBit-Ethernet-Adapter. Betriebssystem: Windows XP SP2). Die Berichte der Verbindungsaktivitäten sind dieser Arbeit in Form von Ethereal-Berichten beigelegt.

[18] Bei einem Update werden ebenfalls *alle* Daten übertragen – Die Rückgabe einer Meldung, wie beispielsweise „Objekt aktuell“, würde bei der Prüfung auf Änderung ausreichen und das zu übertragende Datenvolumen stark verkleinern.

lassen. Wenn viele Prozesse abgearbeitet werden müssen, ist die Kapazitätsgrenze früh erreicht – besonders im Übertragungsmedium gibt es früh Engpässe (ganz abgesehen von den Kosten, die durch einen solchen Traffic entstehen können).

Web Services charakterisiert im Gegensatz zu OAI-PMH ein relativ hoher Traffic *pro Objekt*. Bei Suchanfragen werden SOAP-Nachrichten formuliert und an die Repositories versendet. Diese Antworten ebenfalls mit voluminösen SOAP-Nachrichten. Von der Verwendung von Übertragungsmedien schmaler Bandbreiten ist bei Web Services also ebenfalls abzuraten, da die Menge der zu übertragenden Daten je Objekt sehr viel größer ist ($\approx$ Faktor 10). Allerdings sieht die Web Service-basierte Architektur vor, nie die gesamte Datenhaltung eines Repository zu übertragen, wie es bei OAI-PMH der Fall ist. Bei einer Web Services-Architektur werden nur die wirklich angeforderten Objekte zurückgeliefert. Somit stellt eine Web Service-orientierte Implementierung eine elegantere Lösung dar.

3.2.2.4 Vollständigkeit der Daten

Es gibt weitere, nachteilige Argumente, die gegen eine Web Service-orientierte Implementierung sprechen: Der Service-orientierte Ansatz setzt voraus, dass alle Repositories bei jeder gestellten Anfrage antworten. Das heißt es darf weder Verbindungsprobleme, noch Ausfälle der Repository-Systeme selbst geben. Ist dies dennoch der Fall, werden unvollständige Ergebnisse einer Suchanfrage an die aufrufenden Client-Anwendung zurückgegeben. Für Wissenschaftler oder Gutachter hat dies vehemente Auswirkungen: Unvollständige Suchergebnisse sind unbrauchbar und können nicht weiterverwendet werden.

OAI-PMH bietet dagegen immer den Datenbestand *aller* angeschlossenen Repositories an, wenngleich die Objekte nicht auf einem aktuellen Stand sind.

3.2.2.5 Integration von vordefinierten Funktionen zum Datenzugriff

OAI bietet standardisierte Funktionen durch welche direkt mit der Datenhaltung kommuniziert werden kann. Die Standardisierung verspricht, dass verschiedene Repositories immer die selben Methoden bereitstellen. Ein Harvester kann sicher sein, dass seine Zugriffsanfragen immer in der gleichen Form beantwortet werden.

Die Web Service-API hingegen (im Hinblick auf Fedora) bietet zwar auch vordefinierte Methoden zum Datenzugriff an, allerdings ist durch diese nicht gewährleistet, dass ein Client, welcher auf ein Fedora-Repository zugreifen kann, auch mit anderen Repositories arbeitet, die eine Web Services-Implementierung vorweisen können. Die Schnittstelle ist lediglich durch ihre Übertragungsmechanismen standardisiert, nicht aber in der Implementierung der Funktionalitäten. So kann jeder Entwickler eines Repository individuelle Funktionen bereitstellen, die sich vollkommen von denen anderer Repositories unterscheiden können.

3.2.3 Auswahl

Nun gilt es zwischen einer Web Service-orientierten und einer OAI-PMH-Implementierung für den Zugriff auf (Meta-)Daten abzuwägen. Es liegt klar auf der Hand: Der Service-orientierte Ansatz bringt aktuellere Daten und verursacht weniger Traffic mit der Option gleichzeitiger Requests. Im Gegenzug allerdings sind durch OAI-PMH immer *alle* Daten verfügbar, selbst wenn ein angeschlossenes Repository nicht erreichbar ist. Des Weiteren gibt es vordefinierte Funktionen zum Datenzugriff, die standardisiert auf Repositories mit OAI-PMH-Schnittstelle angewendet werden können.

Der entscheidende Faktor für die Realisierung von Institutionellen Repositories ist für die Zwecke am Alfred-Wegener-Institut allerdings die Skalierbarkeit. Sicherlich ist die bereits seit Jahren eingesetzte Technologie OAI-PMH ein momentanes Mittel zum Zweck, wird aber mit der Zeit, in der der Output an Daten durchgängig steigen wird, schnell an Grenzen stoßen, die mit einer Web Service-orientierten Architektur – wenn auch noch nicht so trivial implementierbar – effizienter gelöst werden kann.

Während OAI-PMH für die Stapelverarbeitung optimiert ist, welche bereits vordefinierte und standardisierte Methoden zum Datenzugriff bereitstellt, bieten Web Services eine Lösung, mit der eine stetig anwachsende Anzahl zu verwaltender Objekte besser handhabbar wird. Dabei ist die Abarbeitung einer Anfrage immer so schnell wie das langsamste Repository – je mehr Repositories zusammengeschlossen werden, desto höher fällt der Zeitvorteil zu OAI-PMH aus.

Die vordefinierten Funktionen bleiben dabei ebenfalls nicht auf der Strecke: Die Funktionalitäten von OAI-PMH sind auch durch Web Services realisierbar. Im Fedora-Repository beispielsweise ist durch die Web Service-Implementierung ein Äquivalent zu OAI-PMH bereits integriert, welches die selben Funktionen beinhaltet und diese sogar durch Funktionen für den Schreibzugriff ergänzt.

Aus diesem Grund soll in dieser Arbeit ein SOAP-Client als Prototyp entwickelt werden, der Web Services-Schnittstellen eines Repository-Systems nutzen kann.

Kapitel 4

Anforderungen an das System

Folgende Anforderungsspezifikation ist bewusst sehr kurz und stichwortartig in Listenform verfasst, damit später bei der Entwicklung der Anwendung die grundlegenden Spezifikationen ohne großen Zeitaufwand nachgeschlagen werden können und somit eine Schnell-Referenz für die einzuhaltenden Richtlinien vorliegt. Die Anforderungsanalyse ist an das Schema der objektorientierten Softwareentwicklung aus [Bal99a] und [Bal99b] angelehnt.

4.1 Zielbestimmung

Ziel der Entwicklung ist die Implementierung eines Such-Clients zur Abfrage von Publikationsdaten aus institutionellen Repositories, welche Informationen über Publikationen in Form von Metadaten bereitstellen.

Der Such-Client greift dabei auf ein externes, bereits lauffähiges Repository (Pangaea) und auf ein internes, prototypisiertes Repository (Fedora) zu.

4.1.1 Muss-Kriterien

Das Produkt der Entwicklung, d.h. die Client-Anwendung, ist nur einsetzbar, wenn es

- Web Service Technologien (SOAP, WSDL) nutzt,
- Mehrbenutzerfähigkeit durch Client-Server-Konzeption aufweist und
- Multithreading-fähig ist.

4.1.2 Kann-Kriterien

Als Kann-Kriterien sind folgende Punkte eventuell zusätzlich zu realisieren:

- Gewährleistung einer hohen Skalierbarkeit zur dynamischen Integration weiterer Repositories via Web Services,
- Dokumentation aller entwickelten Module zur zukunftssicheren Integration in bestehende Geschäftsprozesse durch institutseigene Softwareentwickler.

4.1.3 Abgrenzungskriterien

Da der Rahmen zur Entwicklung der Anwendung zeitlich begrenzt ist, sollen folgende Kriterien bewusst *nicht* erreicht werden, auch wenn sie in vergleichbaren Anwendungen Verwendung finden können:

- Es soll keine hoch-performante Suchmaschine mit Lastverteilung oder verteilter Datenbank entwickelt werden.
- Allerdings soll in diesem Zuge kein Produkt entstehen, welches den Muss-Kriterien nur gerade hinreichend entspricht: Es ist zu berücksichtigen, dass eine Nutzung für den Produktivbetrieb angestrebt wird.

4.2 Einsatz

4.2.1 Anwendungsbereiche

Institutsinterne, sowie externe (öffentliche) Recherche institutsübergreifender Publikationen.

4.2.2 Zielgruppen

- Wissenschaftler
- wissenschaftlich interessierte Personen
- Gutachter

4.2.3 Betriebsbedingungen

- Physikalische Umgebung des Softwaresystems ist eine Unix-Plattform mit Lese-, Schreib- und Ausführungsrechten in mindestens einem Ordner.
- Die Betriebszeit des Produkts ist durchgängig, zu Wartungsarbeiten kann es deaktiviert werden.
- Eine tägliche Beobachtung des Softwaresystems durch einen Administrator wird angeraten, um einen möglichst störungsfreien Betrieb zu gewährleisten.

4.3 Umgebung

4.3.1 Software

Für den Betrieb des Produktes sind folgende Komponenten und Module in den definierten Versionen oder deren Nachfolgeversionen notwendig:

- Java Software Development Kit, Version 1.4
- MySQL, Version 4.0 (oder höher)
- Apache HTTP Server, Version 2.0 (oder höher)
- PHP, Version 5.0 (oder höher), integriert als Modul im Apache HTTP Server

4.3.2 Hardware

Da eine Verwendung für den Produktivbetrieb vorgesehen ist, empfiehlt sich ein möglichst stabiles System, mit genügend Ressourcen zur Verarbeitung vieler Anfragen in kurzen Zeitabständen.

Das zur Verfügung stehende SUNFIRE 15000-System der Firma SUN MICROSYSTEMS mit der eingerichteten Domain „`web`“ (ausgestattet mit 4 Prozessoren á 900 MHz und 8 GB Arbeitsspeicher) sollte zum reibungslosen Betrieb der Server-Anwendungen ausreichen. Falls jedoch Engpässe auftreten sollten, muss die Domain gegebenenfalls erweitert werden – die Plattform bietet mit insgesamt 64 UltraSPARC III Cu Prozessoren (á 900 MHz) und 128 GB Arbeitsspeicher genügend optionalen Spielraum, der zur Zeit nur bedingt genutzt wird.

4.3.3 Orgware

Folgende Vorarbeiten müssen vor der Entwicklung der Anwendung durchgeführt werden:

- Stellung eines Antrags für die Nutzung der Unix-Welt
- Stellung eines Antrags auf eine MySQL-Datenbank
- Aufsetzen eines Fedora-Repository in der Version 2.0
- Importieren von Publikationsdaten bestehender Publikationsdatenbanken in das Repository

4.4 Funktionalität

Die gesamte Funktionalität des Produkts beschränkt sich auf die Suche von Publikationen. Die Besonderheit ist, dass dabei institutseigene, sowie externe Publikationen gefunden werden können.

Der Benutzer der Anwendung formuliert dazu eine Anfrage mit Suchbegriffen mit Hilfe der Client-Anwendung, welches die Anfrage der Suche organisiert an verschiedene Repositories sendet. Die Antworten derer werden mit navigationsunterstützenden Werkzeugen aufbereitet und dem Benutzer angezeigt.

Weiterhin soll der Benutzer Informationen über Funktionalität, Nutzung, Sinn und Betreiber der Webanwendung einholen können. Damit inbegriffen ist die Möglichkeit zur Kontaktaufnahme mit den Verantwortlichen der Anwendung.

4.5 Daten

Die langfristig zu speichernden Daten durch die angeschlossenen Repositories und deren voraussichtlicher Umfang sind in folgenden Abschätzungen dargestellt:

Fedora-Repository:

- aktueller Bestand: ca. 13.000 Publikationen
- jährliche Bestandsänderung: ca. 2.000 hinzukommende Publikationen

Pangaea-Repository:

- aktueller Bestand: ca. 240.000 Datensätze

- jährliche Bestandsänderung: ca. 30.000 hinzukommende Datensätze

Oben genannter Datenumfang und Teilmengen davon müssen durch die zu entwickelnde Anwendung zwar nicht verwaltet werden, allerdings sollen die Abfragemechanismen der Anwendung mit diesen Datenmengen im Lesezugriff umgehen können.

4.6 Leistungen

Bei dem Endprodukt steht die Leistung besonders im Vordergrund. Die linear steigende Menge der zu archivierenden Daten soll mit einem nicht-linear steigenden Suchaufwand gefunden werden können. Insgesamt darf die Anfrage einer Suche nicht länger als 5 Sekunden dauern: Eine Indexierung von Meta-Daten ist unumgänglich. Falls dieses nicht ausreichen sollte, müssen weitere Überlegungen diesbezüglich angestellt werden, falls das Produkt in den Produktivbetrieb übernommen werden soll.

4.7 Benutzeroberfläche

Für alle Elemente der Benutzeroberfläche ist die Sprache *Englisch* zu verwenden, da die Anwendung international verwendbar sein soll.

Die Benutzeroberfläche ist für alle Benutzergruppen gleich. Sie besteht aus einem Formular, in welches ein Suchwort eingegeben werden kann. Mit einem Mausklick auf einen Button mit der Aufschrift „Search“ wird die Suche gestartet.

Nachdem eine Suche abgeschlossen ist, werden Suchergebnisse in verkürzter Form angezeigt (*title*, *author*, *year*). Die Liste der Ergebnisse ist klar strukturiert und kann mittels navigationsunterstützender Elemente übersichtlich beblättert werden. Je Ergebnisseite sollen nicht mehr als zehn Ergebnisse angezeigt werden. Ein Klick auf den Titel eines Eintrags öffnet eine Detailseite, welche alle zur Verfügung stehenden Felder des gewählten Eintrags anzeigt.

Auf der Suchseite, sowie der Ergebnisseite findet jeder Benutzer eine Navigationsleiste mit unterstützenden Links. Diese bieten Hilfestellung und vereinfachen die Navigation innerhalb der Benutzeroberfläche (*start*, *links*, *about*, *help*).

Die genaue Spezifikation der Benutzeroberfläche erfolgt durch den Prototyp in Kapitel 5.2 ab Seite 69.

4.8 Qualitätsziele

4.8.1 Geschwindigkeit

Besonders soll bei der Entwicklung der Anwendung auf Performanz geachtet werden. Das Produkt muss die in 4.6 definierten Leistungen in vollem Umfang erfüllen.

4.8.2 Skalierbarkeit

Die Anwendung sollte darauf ausgerichtet werden, dass ohne größeren Programmieraufwand Repositories dynamisch hinzugefügt, bzw. entfernt werden können: Es sollte ohne Beschränkung auf eine beliebige Anzahl von Repositories skalierbar sein und diese durch Suchanfragen verarbeiten können.

4.8.3 Wartbarkeit / Änderbarkeit

Ebenfalls muss der entwickelte Quellcode zu weiterführenden Verwendungen gut verständlich dokumentiert werden. Nur so kann gewährleistet werden, dass das Produkt zukünftig eingesetzt werden kann.

Zur Dokumentation der Quelltexte soll nach dem PEAR-Standard[1] zur Anwendungsdokumentation vorgegangen werden.

[1] s. [CJM05, Chapter 4]

Kapitel 5

Entwicklung eines Prototypen

5.1 Grundkonzeption

Im Folgenden wird die schrittweise durchgeführte Konzeptionierung und prototypische Entwicklung des zu entwickelnden Prototyps erläutert. Dabei werden verschiedene Richtlinien zum Entwurf von Benutzerschnittstellen berücksichtigt.

5.1.1 Nutzungsszenarien

Die Webseiten zur Suche von Publikationen stellen für den Benutzer folgende Nutzungsszenarien zur Verfügung. Bei allen Szenarien ist der Ausgangspunkt für den Nutzer die Startseite.

5.1.1.1 Suchanfrage und Detailansicht

Der Nutzer befindet sich auf der Startseite und möchte eine Suchanfrage stellen. Dazu gibt er in das Suchformular einen Such-String ein und aktiviert die Kontrollkästchen für die zu durchsuchenden Repositories. Danach klickt er auf die Schaltfläche „Search“. Nachdem die Anfrage von den Repositories verarbeitet wurden, antworten diese auf seine Anfrage: Eine Liste mit den Ergebnissen der eingegebenen Suchparameter wird auf einer neuen Seite angezeigt, die sich im selben Browserfenster öffnet.

Die kompakte Anzeige der Ergebnisse in der Liste reicht dem Nutzer nicht aus, er möchte weitere Information zu einem bestimmten Eintrag bekommen. Dazu klickt er auf den Titel eines Eintrags, welcher mit einem Link verknüpft ist. Das Resultat ist die detaillierte Anzeige zu dem gewählten Eintrag aus der Liste welche sich im selben Browserfenster öffnet. Hier werden nun alle Informationen zu dem gewählten Eintrag übersichtlich aufbereitet dargestellt. Verknüpfungen, wie Links, werden automatisch

mit dem Ziel der Verknüpfung verbunden, so dass der Nutzer direkt auf die Verweise des Eintrags zugreifen kann (Beispiel: DOI[1] wird mit Link zur Quelle versehen).

5.1.1.2 Hilfestellung zur Nutzung

Der Nutzer befindet sich auf der Startseite und möchte Informationen zu der Anwendung einholen. Dazu klickt er auf „help“ in der Menüleiste. Es öffnet sich darauf eine Seite im selben Browserfenster, welche Informationen zur Nutzung und zur verwendeten Technologie der Webanwendung enthält. Die Informationen sind übersichtlich strukturiert und enthalten zum besseren Verständnis nur wenige Fachbegriffe.

5.1.1.3 Kontaktaufnahme

Der Nutzer befindet sich auf der Startseite und möchte sich mit dem Betreiber der Anwendung in Verbindung setzen. Dazu klickt er auf den Menüpunkt „about“. Daraufhin öffnet sich eine Seite im selben Fenster mit Informationen zu den Verantwortlichen mit folgenden Angaben: Namen, Titel, Adressen, Telefonnummern, Email-Adressen.

Zur elektronischen Kontaktaufnahme wählt der Nutzer eine Email-Adresse per Mausklick aus. Da diese bereits mit *mailto*-URL mit dem Email-Programm des Nutzers verknüpft ist, beginnt er direkt nach dem Klick mit dem Schreiben der Nachricht und sendet diese über sein eigenes Email-Programm ab.

5.1.2 Beschreibung des zukünftigen Nutzers

5.1.2.1 Allgemein

Jeder Benutzer hat individuelle Fertigkeiten, Fähigkeiten und Kenntnisse. Es ist nicht möglich, Benutzungsschnittstellen zu realisieren, die allen Benutzern gerecht werden. Eine sinnvolle Vorgehensweise weitgehend benutzergerechte Systeme zu entwickeln ohne jeden einzelnen Benutzer berücksichtigen zu müssen, ist die Benutzer in Benutzerklassen einzuordnen.

5.1.2.2 Benutzergruppen

Die Umgebung in der die Anwendung eingesetzt werden soll gibt die Richtung vor, für welche Benutzergruppen die Anwendung maßgeblich zugeschnitten werden muss.

Aus Erfahrungen mit bestehenden Publikationssystemen geht hervor, dass die Anwendung hauptsächlich im internen Gebrauch durch *Wissenschaftler und Forscher*

[1] DOI: „Digital Object Identifier“ – Bezeichner für digitale Objekte, der eine eindeutige und permanente Identifikation digitaler Objekte möglich macht (mit ISBN oder ISSN vergleichbar).

genutzt wird. Diese Benutzergruppe macht 80% der Gesamtnutzer aus. Die restlichen 20% Nutzer setzen sich aus wissenschaftlich Interessierten, Gutachtern, technischem Personal und sonstigen Benutzern zusammen. Die Anwendung soll dadurch grundlegend auf die Bedürfnisse für das Recherchieren wissenschaftlicher Publikationen ausgelegt sein.

In der folgenden Tabelle sind die zu erwartenden Benutzergruppen mit ihren Fähigkeiten aufgelistet.

Benutzergruppe	EDV-Kenntnisse	Fachkenntnisse	Körperliche und geistige Eigenschaften	Anteil
Wissenschaftler und Forscher	Befriedigend	Sehr Gut	Gut	80%
Sonstige	Ausreichend	Ausreichend	Gut	20%

Tabelle 5.1: Benutzergruppen

5.1.2.3 Häufigkeit der Benutzung

Weitere Untersuchungen ergaben, dass Benutzer von Publikationssystemen in regelmäßigen Abständen Anwendungen dieser Art nutzen. Dadurch ist zu erwarten, dass Benutzer häufig mit der selben Anwendung arbeiten. Daraus schlussfolgert, dass bei fast allen Benutzergruppen eine häufige Benutzung von ähnlichen Anwendungen stattfinden wird. Der Lerneffekt der Benutzer bei der Bedienung des Systems wird hoch ausfallen.

5.1.2.4 Konsequenzen für den Entwurf der Client-Anwendung

Wie unter Punkt 5.1.2.2 geschildert ist zu erwarten, dass 80% über lediglich befriedigende EDV-Kenntnisse verfügen, daraus folgt, dass das Buchungssystem intuitiv zu bedienen sein muss. Klare Strukturen des Systems müssen gegeben sein. Ein aussagekräftiges Hilfesystem ist zu implementieren.

Begrifflichkeiten sind eindeutig zu wählen, auf Begriffe die aus der EDV-Welt stammen sollte, wenn möglich, ganz verzichtet werden (z.B. „URL“).

Zusätzlich muss eine hohe Fehlertoleranz der Anwendung gegeben sein. Die Nutzung durch Gutachter ist dabei besonders zu berücksichtigen: Fehlinformationen und falsche Ergebnis-Ausgaben resultieren in unbrauchbaren Suchergebnissen. Dieses kann bei der Benutzergruppe *Wissenschafter und Forscher* ebenso Grund für eine zukünftige Nicht-Nutzung der Anwendung sein – die Anwendung wäre dann überflüssig.

Allerdings ist zu berücksichtigen, dass die Aufgabenstellung dieser Diplomarbeit lediglich die Entwicklung eines Prototyps vorsieht. Der Schwerpunkt liegt also auf

der grundlegenden Entwicklung eines lauffähigen Prototyps und nicht auf der Entwicklung eines Produktivbetrieb-fähigen Anwendungsystems. Die oben erwähnten Konsequenzen zum Entwurf der Client-Anwendung sind als zu erreichende Ziele für die zukünftige Entwicklung der Software am AWI anzustreben und können, falls noch nicht implementiert, nach dem Abschluss dieser Diplomarbeit umgesetzt werden. Damit das Neben-Ziel der Anwendung (nämlich die Entwicklung einer neuartigen, zukunftsorientierten Anwendung) nicht aus den Augen verloren wird, sind die vorangegangenen Anforderungen dennoch aufgeführt worden.

5.1.3 Grundsätzliche Überlegungen zu Ein- und Ausgabemedien

5.1.3.1 Eingabemedien in Bezug auf zukünftige Nutzer

Die Anwendung richtet sich hauptsächlich an Nutzer, die einer aus Punkt 5.1.2.2 aufgeführten Benutzergruppe zugeordnet werden können.

Die Benutzer der Anwendung werden über einen Standard-Client (in Form eines Desktop-PCs oder Notebooks) verfügen. Die zu erwartenden Eingabemedien werden daher *Tastatur* und *Maus* sein.

Aufgrund der festgelegten Benutzergruppen ist zu erwarten, dass nur ein sehr geringer Prozentsatz die Eingabe der Daten mit einem mobilen Client (PDA) oder über einen Touchscreen vornehmen wird. Aus diesem Grund wird diese Eingabemöglichkeit nicht berücksichtigt.

5.1.3.2 Ausgabemedium

Wie im vorigen Punkt erwähnt, verfügen die Benutzer über einen Standard Client in Form eines Desktop-PCs oder eines Notebooks. Als Ausgabemedium wird daher ein 15” bzw. 17” Röhrenmonitor für Desktop-PCs und ein 14.1” bzw. 15.1” für Notebooks vorausgesetzt. Die Auflösung des Ausgabemediums beträgt im schlechtesten Fall 800 x 600 Pixel. Optimal ist eine Auflösung von 1024 x 768 Pixel.

5.1.3.3 Technologische Grundentscheidungen

Bei den technologischen Grundentscheidungen wird zwischen zwei unterschiedlichen Szenarien unterschieden:

a) Voraussetzungen an den Client (Hard- und Software des Benutzers)

b) Anforderungen an den Server

Um eine möglichst plattformunabhängige Lösung zu erhalten wird die Anwendung über das Internet als Webanwendung realisiert, die im Hintergrund mit den angeschlossenen Repositories via Web Service verbunden ist.

Die grafische Oberfläche ist in PHP programmiert und wird dem Benutzer in Kombination von XHTML und CSS zur Verfügung gestellt.

Daraus ergeben sich sowohl für den Client als auch für den Server folgende Anforderungen:

a) **Voraussetzungen für den Client**

- Standard Client (Desktop oder Notebook)
- Betriebssystem: Windows 9x, NT, 2000, XP, Linux, Solaris, MacOS
- Web Browser (Mindestanforderungen: Internet Explorer 5.5, Mozilla 1.3, Firefox 0.8, Opera 7.0, Netscape Communicator 6, Konqueror 3.3 oder Safari 1.0)
- Internetzugang via Modem oder höher

b) **Anforderungen an den Server**

- Webserver: Apache Web Server 2 mit PHP5-Unterstützung
- Betriebssystem: Windows NT, 2000, XP, Linux, Solaris
- Internetanbindung: T1 oder höher

5.1.3.4 Pflegekonzept

Nach Implementierung der Software werden die Systemadministratoren von den Entwicklern entsprechend geschult, so dass die Pflege der Software von den Administratoren vorgenommen werden kann. Die Pflege beinhaltet vor allem die vollständige Dokumentation von Änderungen innerhalb der Software, sowie regelmäßige Sicherheitskopien aller Quelltexte.

5.1.4 Zusammenstellung der Funktionen

Im Folgenden werden die Aufgaben der Software und deren Programmablauf in Form von ereignisgesteuerten Prozessketten dargestellt. Die Prozessketten orientieren sich an die unter Punkt 5.1.1 erwähnten Nutzungsszenarien und die unter Punkt 5.1.4 aufgeführten Funktionalitäten der Software.

Die Anzahl der Funktionalitäten ist bewusst klein gehalten worden, damit die Programmhandhabung so intuitiv wie möglich wird:

- Suchen und Anzeigen von Metadaten von Publikationen
- Darstellen von Ergebnissen in Detailansicht
- Anzeigen von Informationen über die Webanwendung
- Anzeigen von Hilfeinformationen
- Kontaktaufnahme mit den Verantwortlichen

Die im Folgenden aufgeführten Prozessketten stellen den detaillierten Ablauf der Anwendungsfälle dar:

5.1.4.1 Suchen und Anzeigen von Metadaten von Publikationen / Darstellen von Ergebnissen in Detailansicht

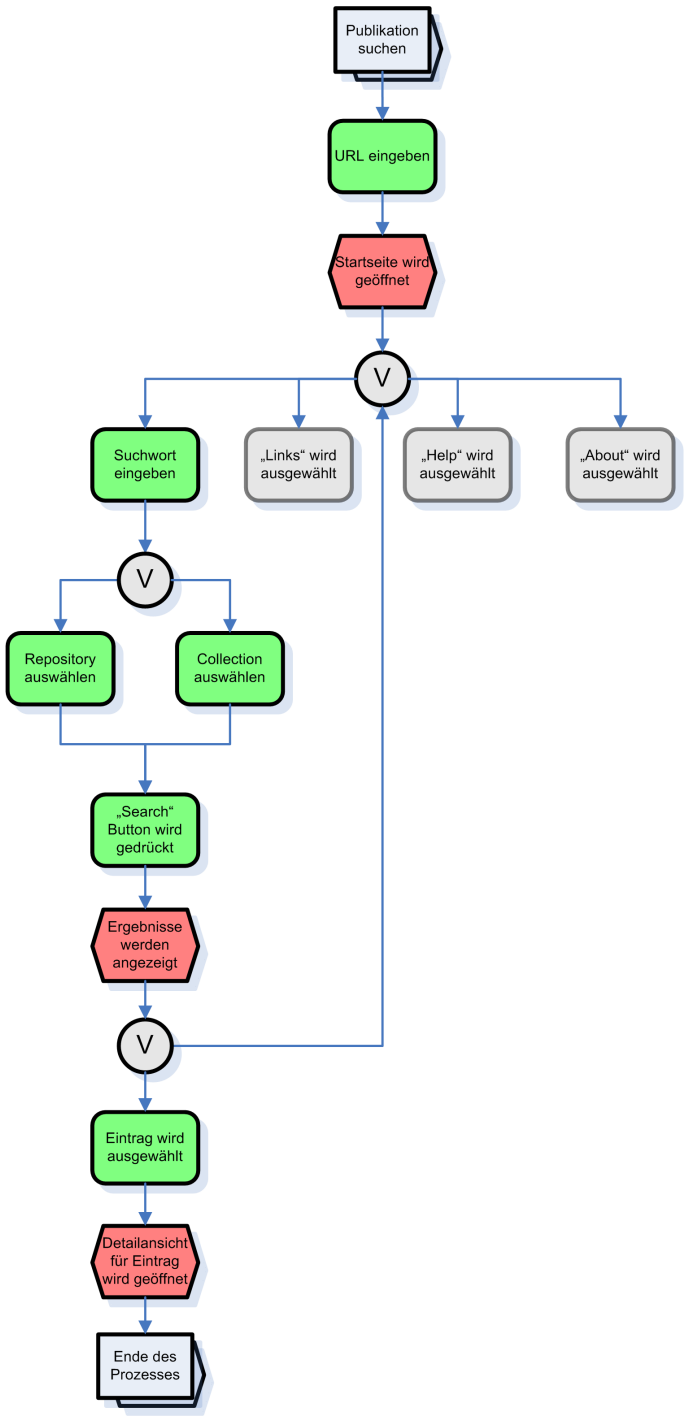


Abbildung 5.1: Anwendungsfall: Publikation suchen

5.1.4.2 Anzeige von Links

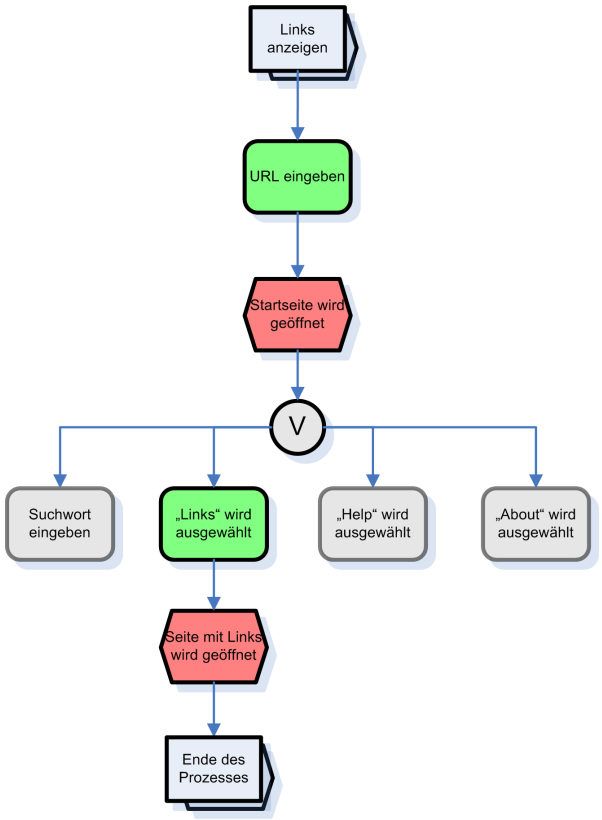


Abbildung 5.2: Anwendungsfall: Links anzeigen

5.1.4.3 Anzeigen von Hilfeinformationen

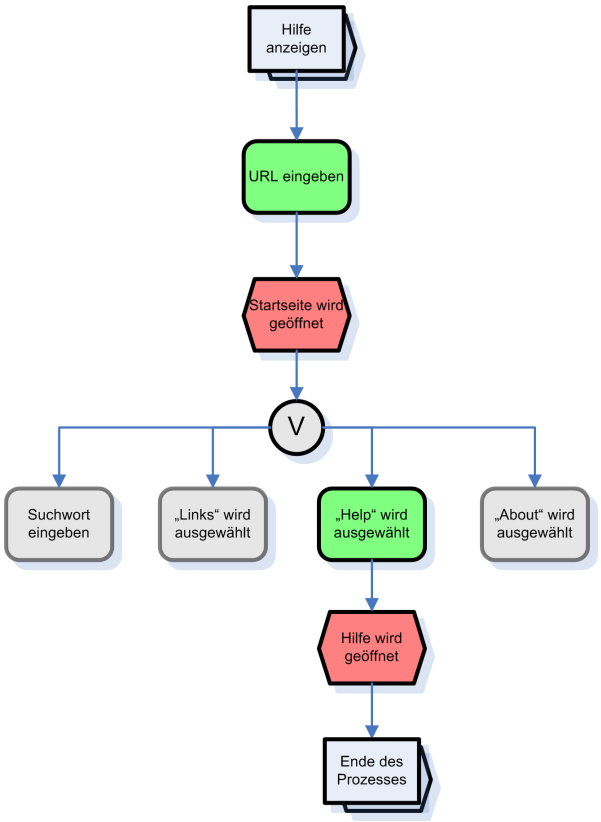


Abbildung 5.3: Anwendungsfall: Hilfe anzeigen

5.1.4.4 Anzeigen von Informationen über die Webanwendung / Kontaktaufnahme mit den Verantwortlichen

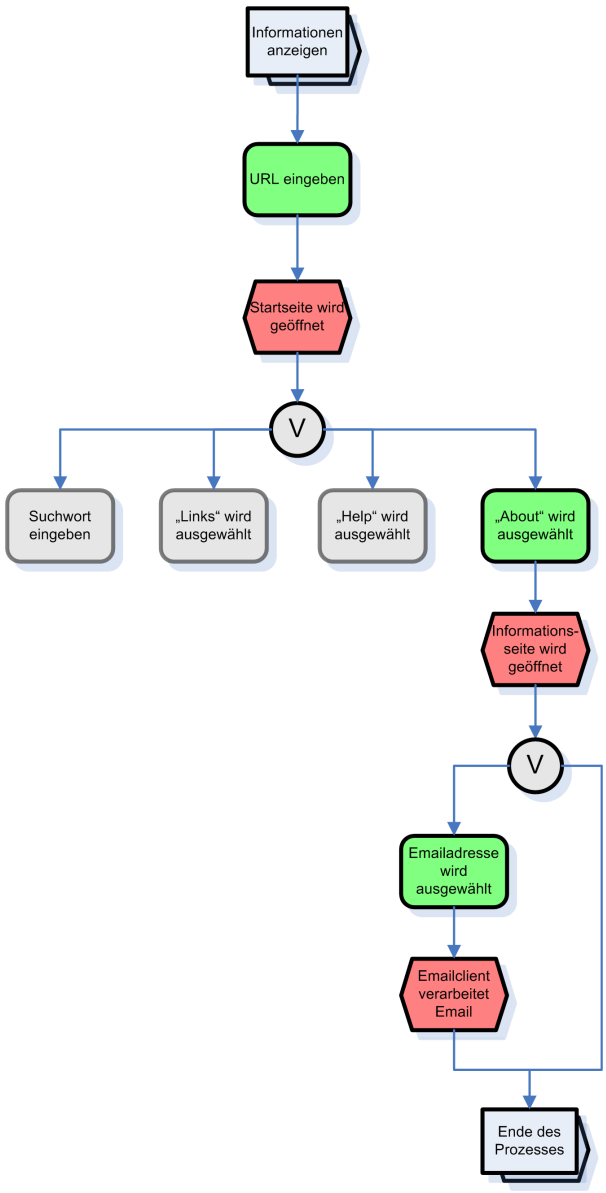


Abbildung 5.4: Anwendungsfall: Informationen anzeigen

5.1.5 Überlegungen zur Eingabe-/Ausgabeschnittstelle

5.1.5.1 Grunddesign

Für den Entwurf der Anwendung liegt ein Grunddesign der Organisation vor. Die Webseiten der Anwendung sind nach dem Schema der vorgegebenen Richtlinien zu erstellen.

5.1.5.2 GUI-Ansatz

Der entwickelte GUI-Ansatz beruht grundlegend auf einem vorgegebenen Konzept, wenngleich dieses nicht festgeschrieben wurde. Unter Berücksichtigung von Richtlinien des Designs bereits bestehender Webanwendungen die das Thema *Open Access* betreffen, wurden die Benutzeroberflächen an die Vorgaben angepasst.

Als Vorlage dazu wurde das Portal der *Arbeitsgemeinschaft Open Access in der Helmholtz Gemeinschaft*[2] genommen, welches ebenfalls am AWI entwickelt wurde. Des Weiteren sind Einflüsse eines weiteren Konzeptes in der Anwendung zu erkennen. Die Vorlage zur Formularentwicklung und zur Ergebnisdarstellung lieferte die Anwendung *PKP-Harvester* des *Public Knowledge Project*[3].

Besondere Bedeutung sind Fragen der Zugreifbarkeit zuzuordnen. Ab 2006 müssen alle Internetauftritte und -angebote sowie öffentlich zugängliche Intranetauftritte und -angebote der Behörden der Bundesverwaltung die 14 Anforderungen des BITV[4] erfüllen. Gerade im Bereich der Webanwendungen gibt es dort noch viele (PHP-basierte) Applikationen, welche heute – ein halbes Jahr vor dem Inkrafttreten der Verordnung – immer noch nicht barrierefrei nutzbar sind. Aus diesem Grund soll bei der Entwicklung des Prototyps und bei der letztendlichen Implementierung besonders Rücksicht auf die Vermeidung von Barrieren genommen werden. Im Rahmen dessen sollen dazu die Richtlinien der Web-Accessibility-Initiative[5] des W3-Consortium Grundlage zur Erstellung der Anwendung bieten, wie durch die Verordnung des BMI[6] vorgeschrieben[7].

5.1.5.3 Grundüberlegungen zu Logo, Farb-, Schrift und Design-Konzept

Das verwendete Logo der entwickelten Anwendung basiert auf bereits existierenden Logos und stellt eine Kombination von Banner der Helmholtz-Gemeinschaft und

[2] s. `http://helmholtz-oai.awi-bremerhaven.de/`
[3] PKP-Harvester wird entwickelt von der University of British Columbia in Vancouver, Canada. Im Internet unter der folgenden Adresse abrufbar: `http://www.pkp.ubc.ca/`
[4] Barrierefreie Informationstechnik-Verordnung, vgl. [BMI02]
[5] s. `http://www.w3.org/WAI/`
[6] Bundesministerium des Innern
[7] vgl. [BMI02, Anlage 1]

der Arbeitsgemeinschaft für Open Access dar. Folgende Grundüberlegungen zur Ein- und Ausgabeschnittstelle mussten selbst erarbeitet werden.

Farbe Durch die Vorgabe der Farben in den Logos und durch die Gestaltung der Portalseiten der Arbeitsgemeinschaft für Open Access ist die Farbgebung der entwickelten Anwendung dieser Diplomarbeit bereits grundsätzlich festgelegt.

Grundsätzlich sind wenige verschiedene Farbtöne verwendet worden, um das Auge des Benutzers nicht unnötig zu überfordern. Die eingesetzten Farben sollen einen Wiedererkennungswert herstellen, um somit eine Identifikation mit dem Institut zu ermöglichen.

Schrift Die ausgewählte Schriftart sollte gradlinig und einem dem Benutzer bekannte Schriftart sein. Die Zeichen müssen bezüglich ihrer Gestalt und Darbietung verwechslungssicher sein. Die Anzahl der Zeichen pro Zeile sollte höchstens 60 Zeichen betragen.

Auch mit der Schrift sollte die Verbindung mit bereits bestehenden Anwendungen des AWI hergestellt werden können. Eine durchgängig gleiche oder vergleichbare Schriftart ist zu verwenden.

Design-Konzept Bei der zu erstellenden Anwendung muss ein durchgängiges Layout zu erkennen sein. Hintergrundfarbe und Schriftarten sind zur Wahrung der Einheitlichkeit auf allen Seiten gleich zu halten. Auf bewegte Bilder und Akustik soll ganz verzichtet werden, der Benutzer sollte nicht abgelenkt werden, sondern sich auf das wesentliche der Anwendung konzentrieren können.

Detaillierte Informationen zu dem umgesetzten Grundlayout-Konzept sind im folgenden Abschnitt aufgeführt.

5.2 Benutzeroberfläche

5.2.1 Farbe und visuelle Kontraste

Für das Grundlayout wurden verschiedene Farbtöne verwendet. Allerdings wurde explizit darauf geachtet, das Farbspektrum so gering wie möglich zu halten. Die unterschiedlich gewählten Farben sollen für den Benutzer eine leicht zu erkennende Struktur widerspiegeln, in der er sich leicht zurechtfinden kann.

Dadurch enthält das Konzept nur vier verschiedene Hauptfarben, in einigen wenigen Abstufungen. Insgesamt finden somit auf der Seite sieben verschiedene, untereinander konsistente Farben Verwendung. Für die einzelnen Seitenelemente wurden die folgenden Farben verwendet (die Hintergrundfarbe der Kästchen repräsentiert jeweils die ausgewählte Farbe):

5.2.1.1 Text

Für eine solide Lesbarkeit wurde die Farbe *Schwarz* für die Darstellung aller Texte gewählt:

Rot: **0%** Grün: **0%** Blau: **0%**

5.2.1.2 Links

Zur Abhebung von Links im Text ist folgende Kontrastfarbe genutzt worden:

Rot: **0%** Grün: **42%** Blau: **65%**

Linkfarbe bei Aktivierung / Hervorhebung durch Mauszeiger:

Rot: **100%** Grün: **100%** Blau: **100%**

Um bereits besuchte Verknüpfungen kenntlich zu machen, wurde eine weitere Farbe zur Markierung von so genannten *visited*-Links eingeführt. Sie dient zur klaren Führung durch die Applikation und zeigt bereits Gesehenes an:

Rot: **47%** Grün: **47%** Blau: **47%**

5.2.1.3 Schaltflächen

Da Schaltflächen eine hohe Aufmerksamkeit erzwingen sollen, da durch sie eine Funktionalität realisiert wird, wird der Schaltflächenhintergrund in einem kräftigen

Blauton dargestellt. Der Schriftzug einer Schaltfläche ist, wie auch hier in diesem Beispiel, weiß eingefärbt:

Rot: **0%** Grün: **20%** Blau: **40%**

5.2.1.4 Hintergrund

Da die Hintergrundfarbe der gesamten Seite einen Großteil der gesamten Anwendung ausmacht, ist besonders bei der Farbgebung darauf geachtet worden, keine „anstrengenden“ Farben zu verwenden – Ein reines Weiß findet durch die seine starke Helligkeit deswegen keine Verwendung. Im Gegensatz dazu wurde hier ein angenehmeres, helles Grau eingesetzt:

Rot: **93%** Grün: **93%** Blau: **93%**

5.2.1.5 Formulare

Zur Absetzung von Formularen vom Hintergrund aus Gründen einer klaren Seitenstrukturierung wurde eine etwas hellere Formularhintergrundfarbe genutzt:

Rot: **97%** Grün: **97%** Blau: **97%**

5.2.2 Schrift

Die gesamte Anwendung ist einheitlich in der selben Schriftart gehalten. Hervorhebungen werden durch Fettschrift, Unterstreichungen oder Veränderung der Schriftgröße realisiert.

5.2.2.1 Schriftart

Durch die zu erreichende Plattformunabhängigkeit ist bewusst auf eine festgelegte Schriftart verzichtet worden. Das Style-Sheet setzt dazu lediglich fest, welche Schriftfamilie genutzt wird. Um ein einheitliches Design mit bestehenden Anwendungen herzustellen ist die verwendete Schriftfamilie eine serifenlose Standardschriftart. Auf MS Windows-Systemen ist dies beispielsweise *MS Sans Serif*.

5.2.2.2 Schriftgrößen

Typ	Größe	Stil
Erste Überschrift	16 Punkt	Fett
Zweite Überschrift	14 Punkt	Fett
Text	12 Punkt	Normal
Verknüpfungen	12 Punkt	Unterstrichen

Tabelle 5.2: Festlegen der Schriftarten

5.2.3 Grundlegendes Seitenkonzept

Das im Folgenden abgebildete Seitenkonzept ergibt sich aus den vorangegangenen Festlegungen und ist bewusst dahingehend entworfen worden, dem Benutzer die Bedienung der Anwendung so einfach wie möglich zu gestalten. Auf Verzierungen akustischer Art und animierte Seitenelemente ist aufgrund dessen ausdrücklich verzichtet worden. Für die Strukturierung der Webseite wurden lediglich Gestaltungselemente textueller Art verwendet:

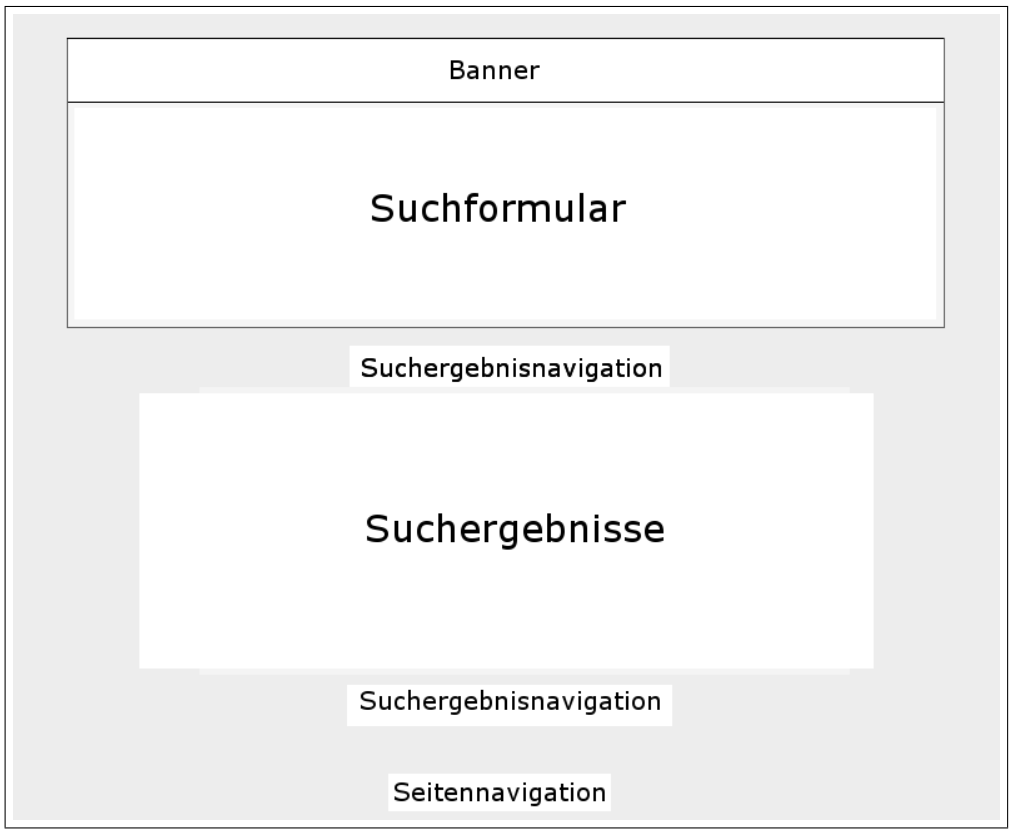


Abbildung 5.5: Grundlayout

5.3 Pflege und Weiterentwicklung

Zur Gewährleistung der Lauffähigkeit der entwickelten Anwendung ist die Festlegung von Maßnahmen zur Pflege erforderlich. Diese sind zur Fortführung der erarbeiteten Ergebnisse notwendig und sollen einen langfristigen Einsatz am Institut sicherstellen.

Da sich der Umfang der entwickelten Anwendung in der Summe auf relativ wenige Zeilen Quellcode erstreckt, ist die Erstellung eines umfangreichen Pflegekonzepts allerdings nicht vorgesehen. Die Pflege ist nach dem Aushändigen der hier erarbeiteten Dokumentation und nach Einweisung der zukünftigen Entwickler in die Anwendungsprogrammierung gewährleistet. Dieses reicht aus, um die Anwendung langfristig einzusetzen und dessen Funktionalitäten zu garantieren.

5.4 Verbreitung

Wichtiges Kriterium ist die Gewährleistung des breiten Einsatzes der Anwendung. Zur Verbreitung soll aus diesem Grund an dieser Stelle darauf hingewiesen werden, dass eine Informationsweitergabe über die Kenntnis der Anwendung an die zukünftigen Nutzer notwendig ist.

Bei der Verbreitung muss allerdings mit Vorsicht vorgegangen werden. Vor einer Bekanntgabe muss sichergestellt sein, dass die Anwendung allen Kriterien zum produktiven Einsatz entspricht. Die Bekanntgabe der Ergebnisse der Entwicklung ist somit erst nach der vollständigen Fertigstellung des Produktes durchzuführen.

Nachdem die notwendigen Funktionen und dessen Fehlerbehandlungen implementiert sind und einer intensiven Prüfung unterzogen wurden, kann mit der Bekanntgabe dessen Adresse und Funktionen durch eine institutsinterne Mitteilung an die Benutzergruppen begonnen werden. Zusätzlich muss eine Verknüpfung im internen Bereich zur Verbreitung erstellt werden, die es den zukünftigen Benutzern erlaubt, direkt auf die Anwendung zuzugreifen. Allerdings ist die Anwendung nicht auf den internen Gebrauch beschränkt: Es müssen Vorkehrungen getroffen werden, um die Anwendung instituts-extern aufrufen und verwenden zu können.

Kapitel 6

Implementierung

6.1 Server-Anwendung

6.1.1 Fedora Repository

Zum Einsatz kommt das Fedora Repository in der Version 2.0. Es wird die vorkompilierte Fassung eingesetzt, die auf der Test-Domain `web` des Sunfire15k-Servers installiert wird. Zur Inbetriebnahme sind grundlegende Überlegungen zur Konfiguration der Anwendung anzustellen.

Die Konfiguration des Fedora Repository wird mittels XML-Datei vorgenommen. Die entsprechenden Parameter werden dazu in der Datei `fedora.fcfg`[1] im `config`-Verzeichnis der Server-Anwendung eingetragen oder verändert. Bei einem Neustart des Systems werden die Einstellungen eingelesen, validiert und gesetzt.

Grundlegende Einstellungen zum Betrieb am AWI sind vorgenommen worden, um das Repository an die bestehende Umgebung anzupassen. Aufgeführt ist im Folgenden eine Auswahl der wichtigsten, grundlegenden Einstellungen:

6.1.1.1 Basiskonfiguration

Name des Repository Der angegebene Name identifiziert das Repository in all seinen Funktionen (z.B. im Web Service). Unter diesem Namen tritt es nach außen hin auf:

```
3  <param name="repositoryName" value="Fedora at AWI"/>
```

Quelltext 6.1: fedora.fcfg: Name des Repository

[1] Die vollständige Konfiguration des Repository-Systems ist in Anhang A.1 ab Seite 137ff. nachzulesen.

Email-Adressen der Administratoren An diese Email-Adresse werden eventuelle Nutzeranfragen gestellt. Sie dient zur Kontaktaufnahme der Betreiber der Repository und wird im Zuge der Nutzung aller von Fedora bereitgestellten Funktionen angezeigt. Hinter der Email-Adresse `fedora-admin@awi-bremerhaven.de` verbirgt sich ein Verteiler, der die Email-Adressen der Administratoren enthält.

```
5  <param name="adminEmailList" value="fedora-admin@awi-bremerhaven.de"/>
```

Quelltext 6.2: fedora.fcfg: Email-Adressen der Administratoren

Speicherort für Objekte Fedora legt zusätzlich zu den Einträgen in der Datenbank die Objektdaten in einer Verzeichnisstruktur an. Dieses Verzeichnis gibt den Ort an, in dem die Objekte gespeichert werden.

```
51  <param name="object_store_base" value="/ext/fedora/data/fedora2_0_objects"/>
```

Quelltext 6.3: fedora.fcfg: Speicherort für Objekte

Temporäres Verzeichnis Zur kurzfristigen Ablage wird dieses Verzeichnis genutzt.

```
56  <param name="temp_store_base" value="/ext/fedora/data/fedora2_0_temp"/>
```

Quelltext 6.4: fedora.fcfg: Temporäres Verzeichnis

Speicherort für Datastreams Ort an dem die Datastreams der Objekte hinterlegt werden. Alle Datastreams, außer des Dublin-Core-Datastreams (DC) werden in diesem Verzeichnis abgelegt.

```
61  <param name="datastream_store_base" value="/ext/fedora/data/fedora2_0_datastreams"/>
```

Quelltext 6.5: fedora.fcfg: Speicherort für Datastreams

Port Der Standard-Port ist der HTTP-Port des integrierten Tomcat-Servers. Er gibt vor, an welchem Port das Repository angesprochen werden kann.

```
81  <param name="fedoraServerPort" value="8080"/>
```

Quelltext 6.6: fedora.fcfg: Port

Adresse des Servers Die absolute Adresse des Repository-Servers wird zur Generierung von URLs genutzt. Beispielsweise werden dadurch auf den Suchseiten interne Links erstellt. Die Adresse `web.awi-bremerhaven.de` zeigt auf die Domain des Servers, auf der Fedora erreichbar ist.

```
87  <param name="fedoraServerHost" value="web.awi-bremerhaven.de"/>
```

Quelltext 6.7: fedora.fcfg: Adresse des Servers

6.1.1.2 Einstellungen zum Objekt-Handling

Besonderes Augenmerk wurde auf die Einstellungen des Objekt-Handling gelegt. Da Veränderungen an diesen (sensiblen) Einstellungen später im Betrieb große Auswirkungen auf den Datenbestand haben können, ist das Verständnis der Einstellungen und dessen Auswirkungen wichtig. Folgende Einstellungen ergaben sich aus wohlüberlegten Entscheidungen zum Objekt-Handling.

Persistent Identifier Namespace Der PID[2] setzt sich aus einer eindeutigen Identifikation, bspw. `Gro2005b`, und einem Namespace zusammen, der die eindeutige Identifikation, die innerhalb einer Institution eindeutig ist, auf eine höhere (weltweite) Ebene erweitern soll. Ein PID im Fedora Repository am AWI hat bspw. folgendes Aussehen: `awi:Gro2005b`. Der PID gibt Aufschluss darüber, woher ein Objekt stammt.

```
126    <param name="pidNamespace" value="awi"/>
```

Quelltext 6.8: fedora.fcfg: Persistent Identifier Namespace

Weitere Namespaces Das Fedora-Repository erlaubt es weitere Namespaces zur Vergabe von PIDs zu verwenden. Mit der Angabe in `retainPIDs` können Weitere genutzt werden. Da sich der hier implementierte Repository-Server im Teststadium befindet, wurde die Einstellung „`*`“ verwendet, um die Möglichkeit offenzuhalten, frei wählbare Namespaces verwenden zu können. Im Produktivbetrieb ist der Einsatz von frei wählbaren Namespaces natürlich durch die Möglichkeit von Mehrfachnennungen kontraproduktiv (Uneindeutigkeit von Objekten) und sollte bis dahin geändert werden.

```
127    <param name="retainPIDs" value="*"/>
```

Quelltext 6.9: fedora.fcfg: Weitere Namespaces

Character-Encoding Diese Einstellung gibt an, in welchem Format die Objekte gespeichert werden sollen. Da die vorhandenen Daten des bestehenden Publikationsservers im UTF-8-Format vorliegen, wurde dieses Format auch für die Objekte des Repository verwendet.

```
129    <param name="storageCharacterEncoding" value="UTF-8"/>
```

Quelltext 6.10: fedora.fcfg: Character-Encoding

[2] *Persistent Identifier*: Eindeutiger Bezeichner für digitale Objekte

6.1.1.3 Datenbankanbindung

Datenbank In `storagePool` wird die Verbindung zur Datenhaltung definiert. `localMySQLPool` gibt an, dass das Modul für den Zugriff auf eine MySQL-Datenbank geladen werden soll.

```
128     <param name="storagePool" value="localMySQLPool" mckoivalue="localMcKoiPool" oraclevalue="
            localOracle9iPool"/>
```

Quelltext 6.11: fedora.fcfg: Datenbank

Verbindung zur Datenbank Da die Verknüpfung mit der MySQL-Datenbank von einem JDBC-Treiber übernommen wird, wird hier die Datenbankverbindung mittels URL konfiguriert. Demnach wird die MySQL-Datenbank auf `mysql://e-db.awi-bremerhaven.de` angesprochen und auf die Datenbank `fedora` zugegriffen. Weitere Einstellungen zur Verbindung sind zusätzlich durch die Angabe von Parametern in der URL angegeben.

```
404     <param name="jdbcURL" value="jdbc:mysql://e-db.awi-bremerhaven.de/fedora?useUnicode=true&amp;
            characterEncoding=UTF-8&amp;autoReconnect=true"/>
```

Quelltext 6.12: fedora.fcfg: Verbindung zur Datenbank

Datenbank-Treiber Der verwendete JDBC-Treiber wird durch die serverseitige MySQL-Distribution bereitgestellt und durch die Angabe folgender Einstellung eingebunden.

```
405     <param name="jdbcDriverClass" value="com.mysql.jdbc.Driver"/>
```

Quelltext 6.13: fedora.fcfg: Datenbank-Treiber

Minimale Verbindungen zur Datenbank Gibt an, wie viele Verbindungen gleichzeitig vorhanden sein müssen.

```
407     <param name="minPoolSize" value="10"/>
```

Quelltext 6.14: fedora.fcfg: Minimale Verbindungen zur Datenbank

Maximale Verbindungen zur Datenbank Analog zur vorangegangenen Option gibt diese an, welche Anzahl gleichzeitiger Verbindungen nicht überschritten werden darf. Da die maximale Anzahl der Verbindungen durch den MySQL-Server vorgegeben wird und jener vorsieht, dass für alle Verbindungen insgesamt „`100`“ nicht überschritten werden darf, wurde hier „`50`“ eingestellt. Dies stellt sicher, dass der Betrieb des Datenbank-Servers nicht durch überflüssige Verbindungsbelegungen beeinträchtigt wird.

```
408     <param name="maxPoolSize" value="50"/>
```

Quelltext 6.15: fedora.fcfg: Maximale Verbindungen zur Datenbank

6.1.1.4 Zugriffsberechtigungen

Zur Sicherheit und aus Gründen der Sicherung von Performanz sind folgende Einstellungen vorgenommen worden.

Hosts Um zu Unterbinden, dass unberechtigte Verbindungen zum Server aufgebaut werden, wurde hier die Loopback-Adresse des Repository-Servers angegeben. Die hier angegebenen IP-Adressen werden allerdings nur dann berücksichtigt, wenn Management-Funktionen aufgerufen werden (Daten löschen / verändern / hinzufügen) – Alle anderen Operationen, die nur Daten abrufen, sind hiervon unberührt; diese Funktionen können von jeder IP-Adresse aufgerufen werden.

```
156    <param name="allowHosts" value="127.0.0.1"/>
```

Quelltext 6.16: fedora.fcfg: Hosts

Maximale Anzahl der Ergebnisse Damit Suchanfragen mit vielen Ergebnissen den Server nicht zum Erliegen bringen, wird die Anzahl der zurückgegebenen Ergebnisse auf „100“ begrenzt. Funktionen zum Abruf von Listen mit den folgenden Ergebnissen werden vom System deswegen bereitgestellt.

```
278    <param name="maxResults" value="100"/>
```

Quelltext 6.17: fedora.fcfg: Maximale Anzahl der Ergebnisse

Sessionlänge Operationen auf dem Repository-Server werden immer einer Session zugeordnet. Die Länge der Session wurde an die der Pangaea-Session angepasst und beträgt somit 180 Sekunden.

```
279    <param name="maxSecondsPerSession" value="180"/>
```

Quelltext 6.18: fedora.fcfg: Sessionlänge

6.1.1.5 Fedora OAI-Servlet

Da in dieser Arbeit besonderes Augenmerk auf die OAI-Fähigkeiten gelegt werden, wurde sehr gewissenhaft bei der Vergabe der OAI-Einstellungen vorgegangen. Nach Möglichkeit wurden alle Einstellungen so vorgenommen, dass sie den Basis-Einstellungen, welche die Web Services bedienen, gleichen.

OAI-Data-Provider Name Name des Repository

```
324    <param name="repositoryName" value="Fedora at AWI"/>
```

Quelltext 6.19: fedora.fcfg: OAI-Data-Provider Name

Domain Über diese Angabe werden die URLs zur Verbindung mit dem Servlet geformt. Sie dienen zur Verknüpfung von Harvestern mit den Daten des Repository.

```
325    <param name="repositoryDomainName" value="awi-bremerhaven.de"/>
```

Quelltext 6.20: fedora.fcfg: Domain

Email-Adresse der Administration Wie in der vorangegangenen Einstellung für das Repository selbst, wurde hier die Verteileradresse der Administration angegeben.

```
326    <param name="adminEmails" value="fedora-admin@awi-bremerhaven.de"/>
```

Quelltext 6.21: fedora.fcfg: Email-Adresse der Administration

Friends Dieser Eintrag enthält BaseURLs von anderen, befreundeten OAI-Data-Provider. Hier wurden die URL's von Pangaea und GKSS angegeben.

```
327    <param name="friends" value="http://ws.pangaea.de/oai/ http://zitmac05.gkss.de/fmi/xsl/oai/oai.
           xsl"/>
```

Quelltext 6.22: fedora.fcfg: Friends

Maximale Anzahl Ergebnisse Spezifiziert die maximale Anzahl von Ergebnissen einer Anfrage.

```
328    <param name="maxRecords" value="100"/>
```

Quelltext 6.23: fedora.fcfg: Maximale Anzahl Ergebnisse

Maximale Anzahl Kopfdaten Analog wird an dieser Stelle die maximale Anzahl der angeforderten Kopfdaten festgelegt.

```
329    <param name="maxHeaders" value="100"/>
```

Quelltext 6.24: fedora.fcfg: Maximale Anzahl Kopfdaten

Lebenszyklus eines Objekts für das Harvesting einstellen Falls ein Objekt im Repository gelöscht werden sollte gibt es drei verschiedene Möglichkeiten für Service Provider, die bereits ein Harvesting durchgeführt haben, bei der Löschung eines Objektes vorzugehen.

Die Einstellung für *deletedRecords* sieht folgende Möglichkeiten vor:

- *no*: Das Repository stellt keine Informationen zu gelöschten Objekten zur Verfügung. Mit dieser Einstellung antwortet ein Repository auf einen GetRecord-Request auf ein gelöschtes Objekt mit einem leeren Response.

- *persistent*: Gelöschte Objekte sind im Repository noch vorhanden, der Status ist allerdings jeweils auf „gelöscht“ geändert. Diese Information wird an den Service-Provider weitergegeben, gelöschte Objekte werden also an den Harvester ausgeliefert.
- *transient*: Mit dieser Einstellung ist nicht garantiert, dass alle Modifikationen an Objekten (Löschen, Erstellen, Ändern) protokolliert wurden. Dem Harvester wird somit vermittelt, dass er bei gelöschten Objekten diese auch aus seinem bereits früher erstellten Harvesting entfernt.

Die bestehende Implementierung des OAI-Provider-Servlets des Fedora Repository wurde mit der Einstellung „*no*“ versehen. In unserem Szenario wird allerdings die „*transient*“-Einstellung durch die Notwendigkeit zur Weitergabe der Löschfunktionalität vorgeschrieben und muss somit geändert werden.

Folgende Zeilen enthalten den zu ändernden Quelltext:

```
136     public DeletedRecordSupport getDeletedRecordSupport() {
137         return DeletedRecordSupport.NO;
138     }
```

Quelltext 6.25: FedoraOAIProvider.java, Zeilen 136-138

Die `FedoraOAIProvider`-Klasse wird von `fedora.oai.*` abgeleitet, wodurch die im Folgenden eingesetzte Konstante `DeletedRecordSupport.TRANSIENT` vererbt wird.

```
136     public DeletedRecordSupport getDeletedRecordSupport() {
137         return DeletedRecordSupport.TRANSIENT;
138     }
```

Quelltext 6.26: FedoraOAIProvider-fixed.java, Zeilen 136-138

Durch die geänderte Einstellung auf *transient* ist gewährleistet, dass ein Harvester gelöschte Objekte ebenfalls nicht weiterhin in seiner Datenbank behält, sondern dass diese ebenfalls entfernt und nicht mehr angezeigt werden.

Herstellen der Kompatibilität Leider entspricht das integrierte OAI-API nicht den Vorgaben der Open Archives Initiative. Dies ergab ein Test, der zur Überprüfung der Kompatibilität des Fedora-Repositorys als OAI-Data-Provider durchgeführt wurde. Die Validierung der API wurde durch die von der OAI zur Verfügung gestellten Tools[3] durchgeführt. Die dabei aufgedeckte Inkompatibilität trat auf, als der Versuch unternommen wurde das Fedora-Repository als Data-Provider im Verzeichnis der OAI einzutragen.

Um als Data-Provider bei der OAI eingetragen zu werden, muss sich ein Repository zertifizieren lassen. Dazu müssen Tests zur OAI-PMH-Konformität durchgeführt

[3] `http://www.openarchives.org/data/registerasprovider.html` – Protocol Conformance Testing

und gänzlich bestanden werden. Leider meistert das Fedora Repository jene im Auslieferungszustand nur bedingt. Dadurch ist es notwendig die im Folgenden erläuterten manuellen Änderungen am Quellcode innerhalb der Fedora-Architektur vorzunehmen. Da Fedora als Open Source daher kommt, ist dies natürlich realisierbar und stellt für den erfahrenen Java-Programmierer kein Problem dar.

Originär besteht die Implementierung der OAI-API aus folgender `getDatePart`-Funktion, die zur Berechnung von Zeitabständen verwendet wird:

```
380     private String getDatePart(Date from, Date until) {
381         if (from==null && until==null) {
382             return "";
383         }
384         StringBuffer out=new StringBuffer();
385         // Note OAI only support ISO8601 dates to the seconds
386         // and Fedora stores dates down to the millisecond level.
387         // This should not matter since OAI requests specify
388         // date ranges (from-until), and as long as the requests
389         // are always in the same units, subsequent requests can pick
390         // up at the last end-point in time (in seconds) without
391         // concern for millisecond granularity.
392         SimpleDateFormat formatter=new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
393         if (from!=null) {
394             out.append(" dcmDate>='");
395             out.append(formatter.format(from));
396             out.append("'");
397         }
398         if (until!=null) {
399             out.append(" dcmDate<='");
400             out.append(formatter.format(until));
401             out.append("'");
402         }
403         return out.toString();
404     }
```

Quelltext 6.27: FedoraOAIProvider.java, Zeilen 380-404

Bei einer *getRecord*-Anfrage an einen OAI-Data-Provider gibt es die Möglichkeit die gestellte Anfrage auf verschiedene Weisen zu spezifizieren. Eine Möglichkeit ist es, die Ergebnisse insoweit einzuschränken, indem ein Bereich eines Zeitabstandes übermittelt wird, und somit lediglich die Ergebnisse zurückgegeben werden, die in dem angegebenen Zeitraum modifiziert, geändert oder erstellt worden sind: Beispielsweise `2005-10-01T15:00:00Z` bis `2005-10-02-T12:30:00Z`. Die Zeitabstände werden dabei bis auf die Sekunden herunter gebrochen und nach dem folgenden, standardisierten Schema[4] angegeben: `JJJJ-MM-TT`*T*`HH:MM:SS`*Z* (Die nicht-schräggestellten Buchstaben gelten als Platzhalter und sind entsprechend zu ersetzen).

Die Verarbeitung von Zeitspannen, scheint in Fedora allerdings ein nicht unerhebliches Problem hervorzurufen. Die Open Archives Initiative gibt vor, dass bei der Abfrage von Zeitabständen der Wertigkeit „0“, mindestens Datensätze mit genau dem Datum zurückgegeben werden, mit dem die übergebene Zeitspanne beginnt. Da Fedora alle Zeitangaben in Millisekunden ablegt, gibt es hier in dieser Funktion eine Unstimmigkeit mit der Konvertierung von Zeitspannen.

[4] vgl. [WW98]

Die Funktion muss daraufhin geändert werden. Zeitabstände mit der Wertigkeit „0“ müssen mindestens den Datensatz zurückliefern, der genau das übergebene Datum *ohne Berücksichtigung* der Millisekunden enthält:

```
380     private String getDatePart(Date from, Date until) {
381         if (from==null && until==null) {
382             return "";
383         }
384         StringBuffer out=new StringBuffer();
385         // Note OAI only support ISO8601 dates to the seconds
386         // and Fedora stores dates down to the millisecond level.
387         // This should not matter since OAI requests specify
388         // date ranges (from-until), and as long as the requests
389         // are always in the same units, subsequent requests can pick
390         // up at the last end-point in time (in seconds) without
391         // concern for millisecond granularity.
392         SimpleDateFormat formatter=new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
393         if (from!=null) {
394             out.append(" dcmDate>='");
395             out.append(formatter.format(from));
396             if (until!=null && from.equals(until)) {
397                 out.append(".000");
398             }
399             out.append("'");
400         }
401         if (until!=null) {
402             out.append(" dcmDate<='");
403             out.append(formatter.format(until));
404             if (from!=null && until.equals(from)) {
405                 out.append(".999");
406             }
407             out.append("'");
408         }
409         return out.toString();
410     }
```

Quelltext 6.28: FedoraOAIProvider-fixed.java, Zeilen 380-410

Erst durch die Änderungen in den Zeilen 396-398 und 404-406[5] ist das Eintragen in die offizielle Liste der Open Archives Initiative als Data-Provider möglich[6]. Doch es sind weitere manuelle Anpassungen an der OAI-API nötig, die allerdings durch lokale Sicherheitsanforderungen notwendig werden und nicht mit der Kompatibilität der OAI-PMH-Schnittstelle zu tun hat (siehe dazu Abschnitt 6.1.2.2 auf Seite 84).

6.1.1.6 Starten des Repository

Setzen der Pfade Bevor der Startvorgang durchgeführt werden kann, ist es erforderlich einige Pfad-Einstellungen zu tätigen. Die Java-Umgebung des Servers erfordert zusätzlich das Setzen von Klassenpfaden, welches aus folgender Abbildung ersichtlich wird. Diese Einstellungen gelten natürlich nur für das hier verwendete System und sind individuell an die Umgebung angepasst.

[5] vgl. Listing 6.27, Seite 80 mit Listing 6.28, Seite 81

[6] s. `http://www.openarchives.org/Register/BrowseSites` – „Fedora at AWI“

```
export JAVA_HOME=/usr/java #PATH TO JAVA 1.4 SDK
export FEDORA_HOME=/ext/fedora #PATH TO FEDORA ROOT FOLDER
export PATH=$PATH:$FEDORA_HOME/server/bin #PATH TO FEDORA BINARY FOLDER
export PATH=$PATH:$FEDORA_HOME/client/bin #PATH TO FEDORA CLIENT BINARY FOLDER
```

Quelltext 6.29: Setzen der Pfade

Start Da nun alle erforderlichen Vorkehrungen zum Start getroffen wurden, kann der Server nun nach der Eingabe von `fedora-start` hochgefahren werden. Wenn alle Einstellungen erfolgreich vorgenommen werden konnten, wird nach der Initiierungsphase „`OK`“ zurückgegeben. Ist dies der Fall, läuft das Repository und kann über die in der `fedora.fcfg`-Datei eingestellte Domain und Port angesprochen werden, siehe Abbildung 6.2.

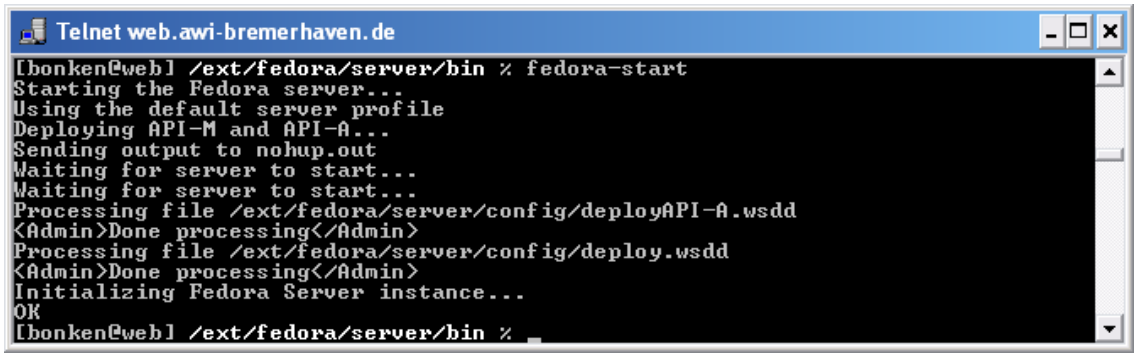


Abbildung 6.1: Shell: fedora-start.sh

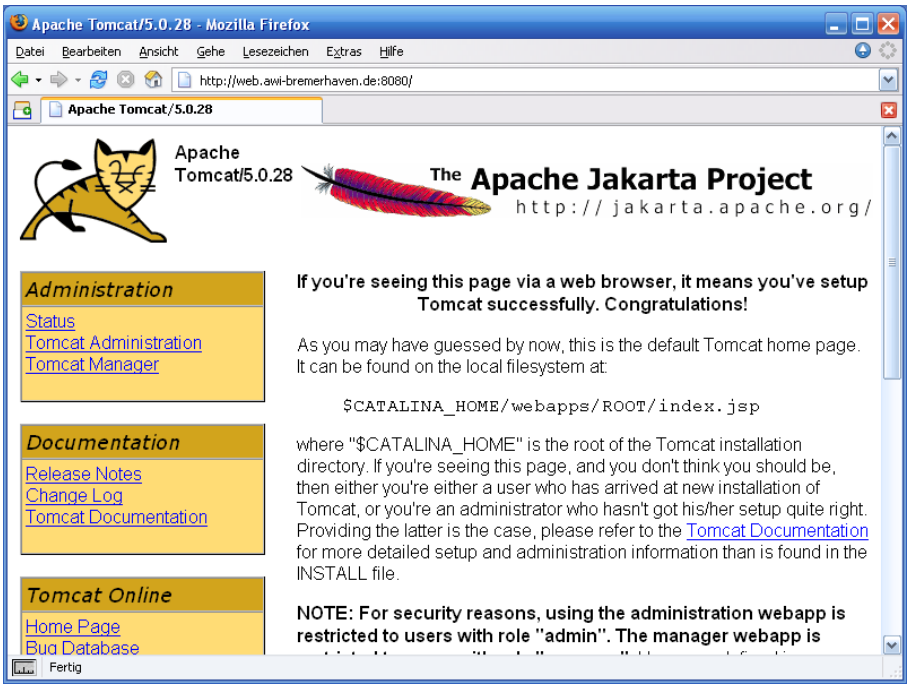


Abbildung 6.2: Startseite des integrierten Tomcat-Applikationsserver

Alle Vorkehrungen sind nun geschaffen, um das Fedora Repository produktiv nutzen zu können. Näheres zu den Zugriffsmöglichkeiten und Schnittstellen des Fedora Repository ist im Abschnitt 6.1.4 auf Seite 88 und in Abschnitt 6.1.5 auf Seite 89 zu finden.

6.1.2 Apache Webserver

Die Laufzeitumgebung muss ebenfalls an die Bedürfnisse des Repository-Systems angepasst werden. Dazu bedarf es weiteren Konfigurationseinstellungen am Web Server, der die Client-Anwendung beherbergen soll.

6.1.2.1 PHP Modul

Zur Ausführung der Client-Anwendung ist das PHP-Modul notwendig. Dieses muss im Apache HTTP Server aktiviert und eingestellt werden. Zusätzlich zur standardmäßigen PHP-Konfiguration im Auslieferungszustand sind folgende Parameter zur Kompilierung zu tätigen, die zum Betrieb der entwickelten Client-Anwendung erforderlich sind. Tabelle 6.1 zeigt die minimal notwendigen Einstellungen – Der Platzhalter *PFAD* ist dabei durch die individuellen Pfade des Systems zu ersetzen (vgl. Listing 6.30):

Integration von PHP in Apache 2	`-with-apxs2=`*PFAD*
Aktivieren des LDAP-Moduls	`-with-ldap`
Authentifikation mit LDAP möglich machen	`-enable-auth-ldap`
XML-Bibliotheken einbinden	`-with-libxml-dir=`*PFAD*
Support zur Konvertierung von Zeichensätzen	`-with-iconv=`*PFAD*
Aktivieren des SOAP-Moduls	`-enable-soap`
Paket zum Handling von URLs	`-with-curl=`*PFAD*
Wrapper für URLs	`-with-curlwrappers`

Tabelle 6.1: PHP Parameter

Der vollständige Befehl zur PHP-Konfiguration mit den notwendigen Parametern auf dem hier verwendeten System hat folgendes Aussehen (Zur Darstellung wurde der Befehl auf mehrere Zeilen umgebrochen):

```
'./configure' '--prefix=/usr/local/php-5.0.4' '--with-apxs2=/usr/local/projects/web/httpd-2.0.54/
    bin/apxs' '--with-mysql=/usr/local/mysql-standard-4.1.10a-sun-solaris2.9-sparc' '--enable-ldap'
     '--enable-auth-ldap' '--with-ldap' '--with-sybase-ct=/opt/csw' '--enable-proxy' '--with-libxml
    -dir=/opt/csw' '--enable-shared' '--enable-pcntl' '--enable-ftp' '--enable-versioning' '--
    enable-session' '--enable-trans-sid' '--with-zlib=/opt/csw' '--enable-mbstring=all' '--with-
    openssl=/opt/csw' '--with-gd' '--with-png-dir=/opt/csw' '--with-jpeg-dir=/opt/csw' '--with-ttf
    =/opt/csw' '--with-iconv=/opt/csw' '--with-iconv-dir=/opt/csw' '--with-xsl=/opt/csw' '--enable-
    soap' '--without-sqlite' '--enable-exif' '--with-curl=/opt/csw' '--with-curlwrappers'
```

Quelltext 6.30: PHP Konfiguration

Die vollständigen Eigenschaften der PHP-Umgebung können ebenfalls unter der URL `http://web.awi-bremerhaven.de/php/phpinfo.php` eingesehen werden.

6.1.2.2 Tomcat-Proxy

Das System ist nur dann einsetzbar, wenn es Zugriffe auf instituts-externe Ressourcen durchführen kann. Das bedeutet, dass die Test-Domain, welche für die Anwendung bereitgestellt wurde, aus Sicherheitsgründen in eine so genannte demilitarisierte Zone (DMZ) verlegt werden muss. Daraus ergeben sich einige Umstände, die es zu berücksichtigen gilt.

Der Auslieferungszustand von Fedora sieht vor, den integrierten Tomcat-Server an den HTTP-Port 8080 zu binden. Da durch die Restriktionen der DMZ dieser Port aber nicht geöffnet werden soll, ist es notwendig dieses Problem mit einer alternativen Methode zu umgehen. Als Lösung wird in diesem Zuge der Einsatz eines Proxy-Servers als geeignet angesehen, der alle Zugriffe auf den Repository-Server über den Port 80 routen soll.

Der verwendete Apache-Server bietet dazu bereits passende Möglichkeiten, um einen Tomcat-Proxy zu implementieren. Folgende Änderungen sind somit an der Web Server Konfiguration nötig:

```
#+++# Connection to tomcat for FEDORA
# 20.07.05, amacario
#
ProxyPass /fedora/oai http://web.awi-bremerhaven.de:8080/fedora/oai
ProxyPassReverse /fedora/oai http://web.awi-bremerhaven.de:8080/fedora/oai
```

Quelltext 6.31: httpd.conf

Obiges Listing zeigt einen Ausschnitt der Konfiguration des Apache HTTP Servers. *ProxyPass* und *ProxyPassReverse* bewirken die Umlenkung von Zugriffen auf die relative URL `/fedora/oai` zur BaseURL des Fedora-OAI-Servlets und dessen zurückgegebenen Antworten. Ein Aufrufer der URL `http://web.awi-bremerhaven.de/fedora/oai` ist somit direkt mit dem OAI-Servlet des entfernten Tomcat-Servers auf Port 8080 verbunden.

Allerdings wirft diese Lösung ein gravierendes Problem innerhalb der Abarbeitung von Fedora's OAI-Funktionen auf. Die interne Angabe der BaseURL bindet das Servlet an die Adresse des Tomcat-Servers mit dem Port 8080, allerdings werden die Antworten des Servlets mit dem Port 80 maskiert. Bei der Validierung des OAI-API durch die Open Archives Initiative scheitert man spätestens daran, dass die BaseURL nicht gleich der URL ist, die aufgerufen wurde (die URL mit Port 80 ist die einzige nach außen Erreichbare, deswegen kann nur diese aufgerufen werden). Aus diesem Grund muss wieder das OAI-Provider Servlet an die lokalen Bedürfnisse angepasst werden.

Die ursprüngliche Zeile in der die BaseURL festgelegt wird hat folgendes Aussehen:

```
79        m_baseURL=baseURL;
```

Quelltext 6.32: FedoraOAIProvider.java, Zeile 79

Ersetzt werden muss diese Zeile durch den folgenden Quelltext, der die BaseURL fest in das Servlet einbindet:

```
79        m_baseURL="http://web.awi-bremerhaven.de/fedora/oai";
```

Quelltext 6.33: FedoraOAIProvider-fixed.java, Zeile 79

Die Lösung mit der Angabe einer „hart-kodierten" URL ist zwar nicht besonders elegant, eine trivialere Lösung ist allerdings ohne die Anpassung zahlreicher Fedora-interner Klassen kaum möglich – der Aufwand stünde in keinem Verhältnis zum Nutzen. Da des Weiteren ein baldiger Releasewechsel ansteht, ist es zudem nicht sinnvoll weitreichende Anpassungen am Quelltext durchzuführen.

6.1.3 Strategie zur Datenübernahme

Zur Übernahme der Publikationen des bestehenden Systems ist die Verlagerung der Datenhaltung durch ein externes Programm notwendig. Da das bereits bestehende System vorerst erhalten bleiben soll und weiterhin produktiv genutzt wird, müssen die Publikationsdaten in regelmäßigen Abständen mit dem Fedora-Repository synchronisiert werden. Das Fedora-Repository tritt dabei allerdings nur als „Kopie“ des bestehenden Systems auf.

6.1.3.1 Ablauf des Imports

Zur Gewährleistung der Synchronisation ist ein Skript entworfen worden, welches jeweils Verbindungen zu dem bereits bestehenden System via LDAP aufbaut und gleichzeitig mit dem Fedora-Repository über SOAP kommuniziert. In Abbildung 6.3 wird die Abarbeitung des Imports anhand des Funktionsablaufs mittels Programmablaufplan erläutert.

6.1.3.2 Implementierung

Ursprünglich wurde die Implementierung eines Import-Skripts im Rahmen dieser Diplomarbeit entwickelt. Der grundlegende Import (mit Einschränkungen) ist in diesem Zuge entwickelt worden. Aufgrund einer Aufgabenumverteilung im Institut wurde die weiterführende Integration des Imports der bestehenden Publikationen in das prototypisierte Fedora-Repository allerdings abgegeben und wird momentan von einer Vollzeitkraft verwaltet und ständig erweitert. Aus diesem Grund wird auf die Erläuterung der Implementierung des Import-Skripts an dieser Stelle verzichtet. Die vorläufige Version des angesprochenen Skripts ist allerdings in Anhang B ab Seite 147 nachzuschlagen.

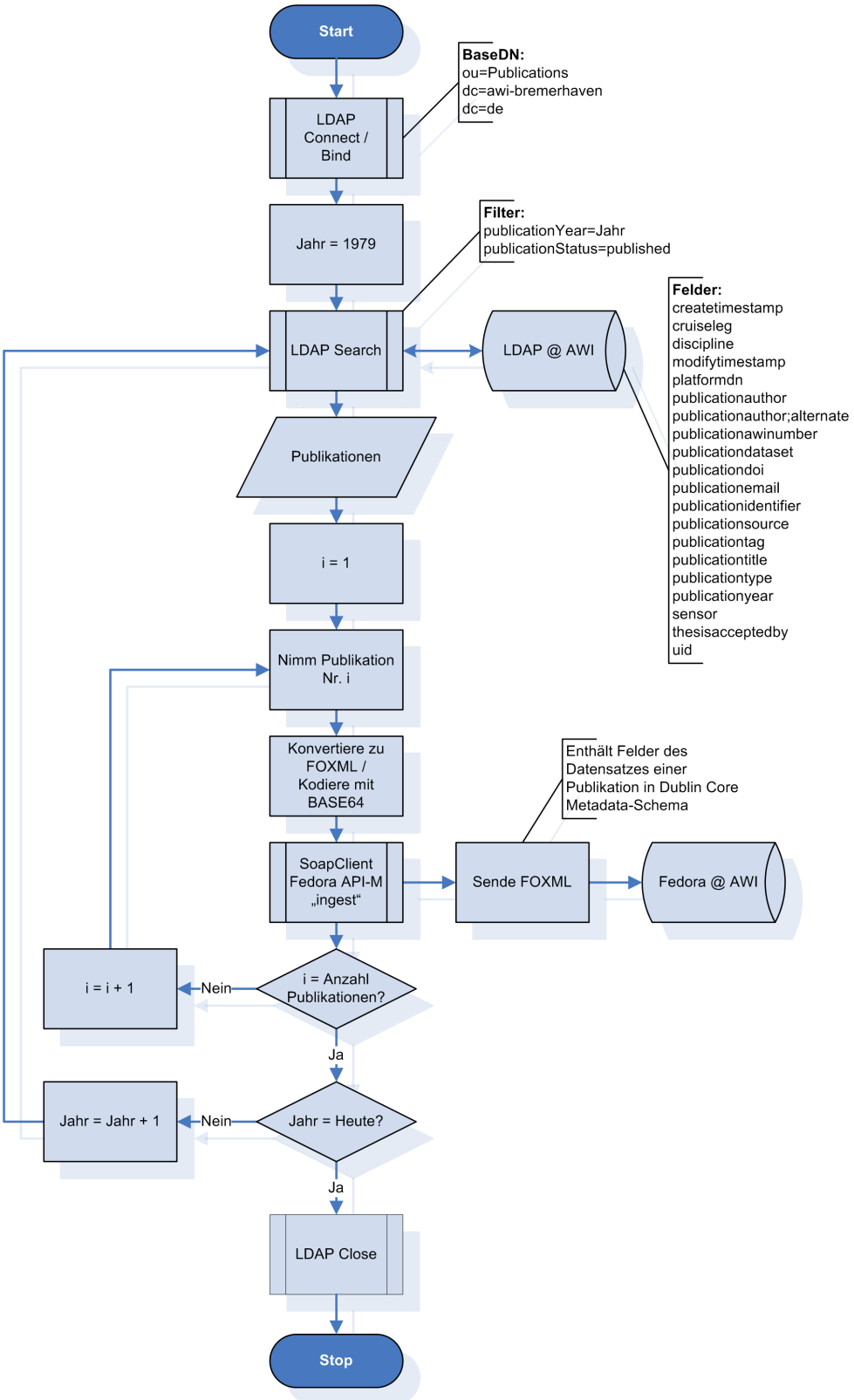


Abbildung 6.3: Programmablaufplan Datenimport

6.1.4 Schnittstellen der Server-Anwendung

6.1.4.1 Fedora OAI-API

Der laufende Server stellt nun zwei Schnittstellen zur Verfügung, über die Operationen auf den Datenbestand erfolgen können. Das OAI-API ist eine davon, welche über das HTTP-Protokoll unter der Adresse `http://web.awi-bremerhaven.de/fedora/oai` abrufbar ist. Diese URL ist gleichzeitig die BaseURL für die OAI-PMH Schnittstelle. Im Folgenden ist der Liste der zur Verfügung stehenden OAI-Funktionen mit Beispielen zum Aufruf aufgeführt[7].

Verb	URL
`Identify`	`http://web.awi-bremerhaven.de/fedora/oai?verb=Identify`
`ListMetadataFormats`	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListMetadataFormats`
`ListSets`	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListSets`
`ListIdentifiers`	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListIdentifiers&metadataPrefix=oai_dc` `http://web.awi-bremerhaven.de/fedora/oai?verb=ListIdentifiers&metadataPrefix=oai_dc&from=2002-02-27`

`ListRecords`	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListRecords&metadataPrefix=oai_dc`
	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListRecords&metadataPrefix=oai_dc&until=2005-12-31`
	`http://web.awi-bremerhaven.de/fedora/oai?verb=ListRecords&metadataPrefix=oai_dc&set=objects`

`GetRecord`	`http://web.awi-bremerhaven.de/fedora/oai?verb=GetRecord&metadataPrefix=oai_dc&identifier=oai:awi-bremerhaven.de:awi:01a2001a`

Tabelle 6.2: Fedora OAI-API

6.1.4.2 Fedora SOAP-API

Die zweite Schnittstelle ist über SOAP erreichbar. Insgesamt gibt es vier Web Services, durch die Zugriffe auf Daten erfolgen können. Wie in Abschnitt 3.1.2.2 bereits erwähnt werden allerdings nur zwei dieser vier im Rahmen dieser Arbeit verwendet. Die Endpunkte der verwendeten Web Services des hier implementierten Servers sind im Folgenden aufgeführt:

- *API-A*:
  `http://web.awi-bremerhaven.de:8080/fedora/access/soap`

[7] siehe auch Abschnitt 3.1.1

- *API-M*:
  `http://web.awi-bremerhaven.de:8080/fedora/management/soap`

Durch das Anhängen des Srings „`?wsdl`“ an die jeweilige URL, wird das zugehörige WSDL-Dokument des Web Services angezeigt[8]; beispielsweise: `http://web.awi-bremerhaven.de:8080/fedora/access/soap?wsdl`.

Zu beachten ist weiterhin, dass die API-M nur per HTTP-Benutzerauthentifizierung durch Angabe von Benutzername und Passwort benutzbar ist. Diese Zugangsdaten sind aus Sicherheitsgründen hier nicht aufgeführt.

6.1.5 Funktionstest

Zur Überprüfung aller Funktionen des Repository ist ein umfangreicher Funktionstest erforderlich, der die Schnittstellen und das Objektmanagement des Fedora-Repository prüft. Dies ist notwendig um qualitativ hochwertige Ergebnisse für den Vergleichstest der in Kapitel 3 erläuterten Architekturen zu erlangen und um Fehler der Konfiguration auszuschließen.

6.1.5.1 Administrationsoberfläche

Zur Überprüfung des Repository wird der mitgelieferte Administrations-Client *Fedora-admin*[9] zu Hilfe genommen, um eine Verbindung zu dem Repository aufzubauen. Dieser kommuniziert ausschließlich über die SOAP-API mit dem Repository, so dass gleichzeitig die Schnittstellen *API-A* und *API-M* in diesem Zuge getestet werden.

Zur Überprüfung der Managementfunktionen (*API-M*) wird ein Testobjekt mit folgendem Inhalt über die bereitgestellte Administrationsoberfläche in das Repository eingefügt. Es hat folgenden Inhalt in Form von Metadaten nach Dublin-Core-Schema.

```
<dc>
 <title>Neue Wege des Publizierens: Die Zitierfähigkeit wissenschaftlicher Primärdaten</title>
 <creator>Brase, J.</creator>
 <creator>Diepenbroek, M.</creator>
 <creator>Grobe, H.</creator>
 <creator>Höck, H.</creator>
 <creator>Klump, J.</creator>
 <creator>Lautenschlager, M.</creator>
 <creator>Schindler, U.</creator>
 <creator>Sens, I.</creator>
 <description>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=Bra2005b</
     description>
 <publisher>11. Jahrestagung IuK (Initiative Information und Kommunikation der wissenschaftlichen
     Fachgesellschaften in Deutschland, Bonn)., 2005</publisher>
```

[8] Übliche Vorgehensweise bei durch Apache-Axis implementierte Web Services

[9] Dieser befindet sich im Verzeichnis `/bin` der Client-Distribution. Das Skript `fedora-admin.sh` bzw. `fedora-admin.bat` startet die in JAVA entwickelte Benutzeroberfläche.

```
13  <date>2005-01-01</date>
14  <type>event, national talk</type>
15  <identifier>Bra2005b</identifier>
16  <identifier>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=Bra2005b</
        identifier>
17  <rights>uri:http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode</rights>
18  </dc>
```

Quelltext 6.34: Testobjekt Bra2005b

Das Einfügen übernimmt die Funktion „New Data Object“. Für das neue Objekt wird als PID „awi:Bra2005b“ vergeben, die Metadaten werden mittels Datastream angehängt. Abbildung 6.4 zeigt das Einfügen durch die Administrationsoberfläche.

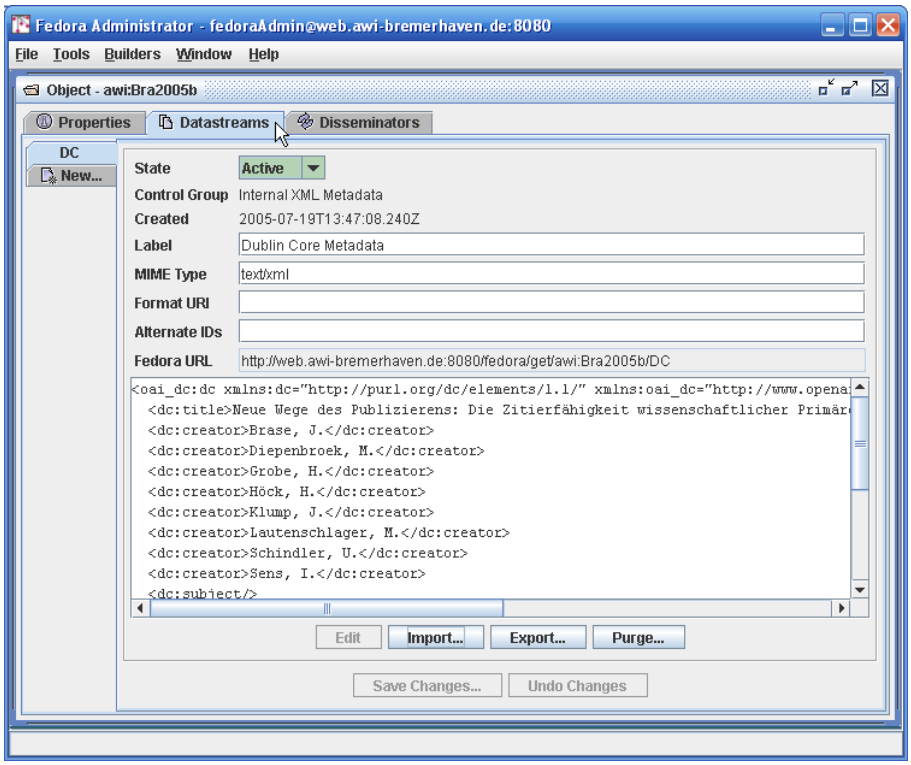


Abbildung 6.4: Einfügen des Objektes durch die Administrationsoberfläche

Die im Folgenden aufgeführte Objektrepräsentation zeigt unter Anderem, dass die Einfügeoperation des durchgeführten Tests Erfolg hatte.

6.1.5.2 Objektrepräsentation in Fedora

Fedora speichert Objekte in einem besonderen Format: FOXML (Fedora Object XML). Die im vorangegangenen Abschnitt aufgeführten Test-Metadaten (Listing 6.34) werden durch Repository-interne Mechanismen in dieses Format konvertiert

und in einer Verzeichnisstruktur[10] als XML-Dokument abgelegt. Gleichzeitig wird ein Eintrag in der angeschlossenen Datenbank (in diesem Fall in MySQL) hinterlegt, welche eine Referenz auf die Datei, die Metadaten und weitere interne Informationen enthält. Listing 6.35 repräsentiert die FOXML-Darstellung des Objektes der oben angegebenen Metadaten nach dem Einfügen in das Repository.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <foxml:digitalObject xmlns:foxml="info:fedora/fedora-system:def/foxml#" xmlns:xsi="http://www.w3.
       org/2001/XMLSchema-instance" xmlns:audit="info:fedora/fedora-system:def/audit#"
       xsi:schemaLocation="info:fedora/fedora-system:def/foxml# http://www.fedora.info/definitions
       /1/0/foxml1-0.xsd" PID="awi:Bra2005b">
3   <foxml:objectProperties>
4    <foxml:property NAME="http://www.w3.org/1999/02/22-rdf-syntax-ns#type" VALUE="FedoraObject"/>
5    <foxml:property NAME="info:fedora/fedora-system:def/model#state" VALUE="Active"/>
6    <foxml:property NAME="info:fedora/fedora-system:def/model#createdDate" VALUE="2005-07-19T13:47:08
         .240Z"/>
7    <foxml:property NAME="info:fedora/fedora-system:def/view#lastModifiedDate" VALUE="2005-07-19
         T13:47:08.289Z"/>
8   </foxml:objectProperties>
9   <foxml:datastream ID="DC" STATE="A" CONTROL_GROUP="X" VERSIONABLE="true">
10   <foxml:datastreamVersion ID="DC1.0" LABEL="Dublin Core Metadata" CREATED="2005-07-19T13:47:08.240Z
         " MIMETYPE="text/xml" SIZE="1277">
11    <foxml:xmlContent>
12     <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" xmlns:dc="http://purl.org/
           dc/elements/1.1/">
13      <dc:title>Neue Wege des Publizierens: Die Zitierfähigkeit wissenschaftlicher Primärdaten</
            dc:title>
14      <dc:creator>Brase, J.</dc:creator>
15      <dc:creator>Diepenbroek, M.</dc:creator>
16      <dc:creator>Grobe, H.</dc:creator>
17      <dc:creator>Höck, H.</dc:creator>
18      <dc:creator>Klump, J.</dc:creator>
19      <dc:creator>Lautenschlager, M.</dc:creator>
20      <dc:creator>Schindler, U.</dc:creator>
21      <dc:creator>Sens, I.</dc:creator>
22      <dc:subject></dc:subject>
23      <dc:description>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=
            Bra2005b</dc:description>
24      <dc:publisher>11. Jahrestagung IuK (Initiative Information und Kommunikation der
            wissenschaftlichen Fachgesellschaften in Deutschland, Bonn)., 2005</dc:publisher>
25      <dc:date>2005-01-01</dc:date>
26      <dc:type>event, national talk</dc:type>
27      <dc:identifier>Bra2005b</dc:identifier>
28      <dc:identifier>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=
            Bra2005b</dc:identifier>
29      <dc:identifier>awi:Bra2005b</dc:identifier>
30      <dc:rights>uri:http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode</dc:rights>
31     </oai_dc:dc>
32    </foxml:xmlContent>
33   </foxml:datastreamVersion>
34  </foxml:datastream>
35 </foxml:digitalObject>
```

Quelltext 6.35: FOXML: awi:Bra2005b

Es fällt auf, dass viele interne Informationen zu den ursprünglichen Metadaten hinzugekommen sind. Detailliertere Informationen dazu sind in der Referenz zu FOXML[11] einsehbar und sollen an dieser Stelle nicht weiter erläutert werden.

[10] Je nach Konfiguration verschieden: siehe dazu Listing 6.1.1.1 auf Seite 74

[11] `http://www.fedora.info/download/2.0/userdocs/digitalobjects/introFOXML.html`

6.1.5.3 SOAP-Kommunikation

Zur Überprüfung der API-A wurde die SOAP-Kommunikation untersucht. Dazu wurde die Methode *findObjects*[12] ausgeführt, welche die Metadaten des weiter oben erstellten Objekts zur Anzeige an einen Client zurückliefert. Dazu wurde ebenfalls die Administrationsoberfläche verwendet. Zur Visualisierung der SOAP-Kommunikation wurde allerdings eine Anwendung zwischengeschaltet, welche den SOAP-Verkehr über einen alternativen Port umleitet und für die Zwecke des Debugging anzeigen kann. Das Programm heißt *Tcpmon* und ist im Lieferumfang von Apache-Axis enthalten.

Tcpmon wird nun so eingestellt, dass alle Anfragen auf die Adresse und den Port des Endpunktes der API-A[13] über `http://localhost:1234` geroutet werden. Dazu muss der Administrations-Client mit der virtuellen Adresse des Tcpmon (also `http://localhost:1234`) verbinden. Alle Anfragen an das Repository, wie auch dessen Antworten, werden nun über den Tcpmon „umgeleitet“ – Die Funktionalitäten bleiben dadurch aber trotzdem erhalten.

Folgendes Listing zeigt den Quelltext eines `findObjects`-Requests mit dem Suchwort „`awi:Bra2005b`“:

Fedora SOAP-Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http://www
    .fedora.info/definitions/1/0/api/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns2="
    http://www.fedora.info/definitions/1/0/types/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="
    http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <ns1:findObjects>
   <resultFields SOAP-ENC:arrayType="xsd:string[5]" xsi:type="ns2:ArrayOfString">
    <item xsi:type="xsd:string">title</item>
    <item xsi:type="xsd:string">creator</item>
    <item xsi:type="xsd:string">date</item>
    <item xsi:type="xsd:string">type</item>
    <item xsi:type="xsd:string">pid</item>
   </resultFields>
  <maxResults xsi:type="xsd:nonNegativeInteger">5</maxResults>
  <query xsi:type="ns2:FieldSearchQuery">
   <conditions xsi:nil="1"/>
    <terms xsi:type="xsd:string">awi:Bra2005b</terms>
   </query>
  </ns1:findObjects>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quelltext 6.36: Fedora SOAP-Request

Die Antwort des Repository beinhaltet die Metadaten des angeforderten Objektes, dies zeigt Listing 6.37:

[12] siehe dazu auch Abschnitt 3.1.2.3, Seite 40
[13] `http://web.awi-bremerhaven.de:8080/fedora/access/soap`

Fedora SOAP-Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:findObjectsResponse xmlns:ns1="http://www.fedora.info/definitions/1/0/api/" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <response href="#id0"/>
  </ns1:findObjectsResponse>
  <multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns2="http://www.fedora.info/definitions/1/0/types/" id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns2:FieldSearchResult">
   <listSession xsi:type="ns2:ListSession" xsi:nil="true"/>
   <resultList xsi:type="soapenc:Array" soapenc:arrayType="ns2:ObjectFields[1]">
    <item href="#id1"/>
   </resultList>
  </multiRef>
  <multiRef xmlns:ns3="http://www.fedora.info/definitions/1/0/types/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id1" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns3:ObjectFields">
   <pid xsi:type="xsd:string">awi:Bra2005b</pid>
   <label xsi:type="xsd:string" xsi:nil="true"/>
   <fType xsi:type="xsd:string" xsi:nil="true"/>
   <cModel xsi:type="xsd:string" xsi:nil="true"/>
   <state xsi:type="xsd:string" xsi:nil="true"/>
   <ownerId xsi:type="xsd:string" xsi:nil="true"/>
   <cDate xsi:type="xsd:string" xsi:nil="true"/>
   <mDate xsi:type="xsd:string" xsi:nil="true"/>
   <dcmDate xsi:type="xsd:string" xsi:nil="true"/>
   <title xsi:type="xsd:string">Neue Wege des Publizierens: Die Zitierfähigkeit wissenschaftlicher Primärdaten</title>
   <creator xsi:type="xsd:string">Brase, J.</creator>
   <creator xsi:type="xsd:string">Diepenbroek, M.</creator>
   <creator xsi:type="xsd:string">Grobe, H.</creator>
   <creator xsi:type="xsd:string">Höck, H.</creator>
   <creator xsi:type="xsd:string">Klump, J.</creator>
   <creator xsi:type="xsd:string">Lautenschlager, M.</creator>
   <creator xsi:type="xsd:string">Schindler, U.</creator>
   <creator xsi:type="xsd:string">Sens, I.</creator>
   <date xsi:type="xsd:string">2005-01-01</date>
   <type xsi:type="xsd:string">event, national talk</type>
  </multiRef>
 </soapenv:Body>
</soapenv:Envelope>
```

Quelltext 6.37: Fedora SOAP-Response

Die Serverimplementierung ist damit abgeschlossen. Nun kann ein Client implementiert werden, der die API's zum Zugriff auf die Datenbestände nutzen kann.

6.2 Client Anwendung

6.2.1 Implementierung der Funktionalitäten

Aus der vorangegangenen Anforderungsanalyse zur Entwicklung des Prototyps ergeben sich die umzusetzenden Funktionalitäten für die Client-Anwendung. Im Folgenden wird auf die mittels PHP realisierten Skripte eingegangen, welche die Anforderungen und Funktionen implementieren. Die Vorlagen zur Entwicklung sind in der Grundkonzeption des Prototyps in Abschnitt 5.1 ab Seite 57 detailliert erläutert. Der Zugriff auf die Webanwendung erfolgt durch den Aufruf des Skriptes `index.php`.

6.2.1.1 Suchen und Anzeigen von Metadaten von Publikationen

Methode: simpleSearch Das Suchen von Metadaten von Publikationen wird durch die Bereitstellung eines XHTML-Formulars realisiert. In jenem Formular können Eigenschaften zur Suche spezifiziert werden, die mittels HTTP-GET an die Funktion `simpleSearch` übermittelt werden. `simpleSearch` ist je Web Service individuell implementiert und übernimmt die Konvertierung der Suchbegriffe in ein dem Web Service entsprechend verständliches Format zur Abarbeitung der Suchanfrage. Der Transport wird mittels SOAP realisiert.

Nachdem eine Anfrage versendet ist, wird gewartet bis eine Antwort eines Repository eintrifft. Überschreitet die Zeit der Antwortreaktion allerdings den durch die Konfigurationsdatei eingestellten Timeout, wird die Anfrage mit einer Fehlermeldung beendet. Ist dies nicht der Fall, werden die Inhalte aus dem SOAP-Response an eine interne Funktion übermittelt, welche die Suchergebnisse in ein vereinheitlichtes XML-Format konvertiert. Erst nach der Umwandlung werden durch `simpleSearch` die Ergebnisse der Suchanfrage an den Web Browser zurückgeliefert und ausgegeben.[14]

Methode: moreResults Zu beachten ist, dass `simpleSearch` immer nur einen Ausschnitt der Treffer einer Suchanfrage zurückliefert. Die Anzahl der Ergebnisse ist durch das jeweilige Repository beschränkt (Fedora-Repository und Pangaea-Repository liefern je Anfrage maximal 100 Ergebnisse). Mit einer weiteren Suchanfrage, können allerdings die nächsten Treffer der Suche angefordert und ausgegeben werden.

Diese Funktionalität ist durch die integrierte Methode `moreResults` implementiert worden. Prinzipiell ähnelt sie der `simpleSearch`-Funktion, sie ist allerdings dafür zuständig, nach der Angabe einer Session-ID (aus einer `simpleSearch`-Anfrage) die

[14] Quelltext zu `simpleSearch` in Anhang C.2, Seite 158ff; Listing C.2.1 (Zeile 90ff) und Listing C.2.2 (Zeile 66ff).

folgenden Ergebnisse zu ermitteln. Diese Methode ist solange ausführbar, bis das Ende der Liste erreicht ist – also solange, bis alle Ergebnisse einer Suchanfrage ausgeliefert wurden.[15]

6.2.1.2 Darstellen von Ergebnissen in Detailansicht

Methode: detailsForObject Beide vorangegangenen Funktionen zur Ergebnisanzeige sind in dem Umfang der Ausgabe beschränkt. Dies ergibt sich aus Gründen zur Wahrung der Übersichtlichkeit. Um detailliertere Informationen eines Objektes zu erlangen, ist die Funktion `detailsForObject` für jeden Web Service individuell implementiert worden. Diese ruft jeweils nur die Daten *eines* Objektes ab.

Die Ergebnisse der Abfrage werden allerdings nach dem selben Schema vereinheitlicht, wie die im vorigen Abschnitt beschriebenen Funktionen `simpleSearch` und `moreResults`. Der einzige Unterschied besteht darin, dass die Rückgabe nur ein einzelnes Objekt enthält. Eine Session-ID ist nicht enthalten.[16]

6.2.1.3 Anzeigen von Hilfe, Links und weiteren Informationen zur Anwendung

Skripte: help.php, links.php, about.php Neben `index.php` bestehen drei weitere Skripte zur Erzeugung von XHTML-Dokumenten. Sie dienen allerdings im Gegensatz zu `index.php` zur Generierung statischer Ausgaben. Während `help.php` dafür zuständig ist, dem Benutzer Hilfeinformationen anzuzeigen, bieten `links.php` und `about.php` allgemeine Informationen zur Anwendung und dessen Verantwortliche. Zur Kontaktaufnahme mit den Verantwortlichen der Anwendung ist `about.php` zusätzlich zuständig.[17]

6.2.2 Konfiguration

Zur Konfiguration der Anwendung wurde eine zentrale Konfigurationsdatei (`config.inc.php`) verwendet, um Redundanzen zu vermeiden. Besonders wurde darauf geachtet, dass die gesamte Konfiguration aller betreffender Skripte, in welchen Einstellungsmöglichkeiten bestehen, auf diese Konfigurationsdatei ausgelagert wird. Der Vorteil besteht darin, dass an zentraler Stelle globale Einstellungen getätigt werden können.

[15] Quelltext zu `moreResults` in Anhang C.2, Seite 158ff; Listing C.2.1 (Zeile 120ff) und Listing C.2.2 (Zeile 103ff).

[16] Quelltext zu `detailsForObject` in Anhang C.2, Seite 158ff; Listing C.2.1 (Zeile 146ff) und Listing C.2.2 (Zeile 134ff).

[17] Quelltext `help.php` in Anhang C.6, Seite 187ff; Listing C.6.8 – Quelltext `links.php` in Anhang C.6, Seite 189ff; Listing C.6.9 – Quelltext `about.php` in Anhang C.6, Seite 190ff; Listing C.6.10.

In der Konfigurationsdatei werden somit die grundlegenden Einstellungen für die Anwendung festgelegt:

- Verwendete Web Services und deren WSDL-Beschreibung,
- Anzahl der Suchergebnisse pro Ergebnisseite und Web Service,
- Einstellungen zur Verbindung mit den Web Service und
- weitere spezifische Parameter der Systemumgebung.

Der Inhalt der vollständigen Konfigurationsdatei ist in Anhang C.5.1 auf Seite 172 zu finden.

6.2.3 Verbindung zur Datenhaltung

Zu jedem Repository muss eine individuelle Klasse zum Zugriff auf die Datenhaltung über den integrierten Web Service entwickelt werden. Die Web Services sind (zumeist) unterschiedlich aufgebaut und bieten verschiedene Methoden an, mit welchen verschiedene Ergebnisse erzielt werden können. Um die Funktionalitäten zur Suche zu vereinheitlichen, müssen durch das Entwerfen von Funktionen mit einheitlichen Übergabeparametern sowie vereinheitlichten Rückgabeparametern erstellt werden.

Zur Erfüllung der Ziele der Prototypenentwicklung sind dazu je Web Service drei globale Funktionen entwickelt worden, die nach einem einheitlichen Schema arbeiten (Auf die einzelnen Funktionen wurde bereits in Punkt 6.2.1 genauer eingegangen):

- *simpleSearch*: Startet eine Suche mit einem übergebenen Suchwort, liefert Ergebnisse XML-kodiert zurück.
- *moreResults*: Führt eine zuvor gestartete Suche fort, liefert Ergebnisse XML-kodiert zurück.
- *detailsForObject*: Zeigt alle Details eines Objektes.

Der Zugriff auf die Datenhaltung wird bei allen genannten Funktion durch die SOAP-Schnittstelle des jeweiligen Repository realisiert. PHP bietet in der Version 5 bereits im Auslieferungszustand die Unterstützung für das Erstellen von Verbindungen via SOAP. Bevor allerdings eine Kommunikation über diese Schnittstelle zu einem Repository hergestellt werden kann, muss die damit in Verbindung stehende WSDL-Beschreibung eingelesen werden.

Folgendes Listing zeigt den Aufbau der *simpleSearch*-Funktion, welche auf das Fedora-SOAP-API zugeschnitten wurde:

```
90  function simpleSearch($terms,$numberOfResults) {
91   try {
92    /* Performing a 'findObjects'-SOAP-request: retrieve WSDL, build query, send SOAP-envelope,
         return response */
93    $soapclient = new SoapClient($this->wsdl,array("trace" => 1,"exceptions" => 1));
94    $query = array("conditions" => null,"terms" => "*".utf8_encode($terms)."*");
95    $response = $soapclient->findObjects($this->fields_thumbnail,$numberOfResults,$query);
96    $this->session = isset($response->listSession->token) ? $response->listSession->token : "";
97    return $this->map($response->resultList);
98   } catch(SoapFault $fault) {
```

Quelltext 6.38: class.fedoraWebService.php, Zeilen 91 - 98

Erster Schritt ist das Erstellen eines SOAP-Clients mit dem Einlesen der WSDL-Beschreibung[18]. Darauf folgt die Erstellung eines Übergabeparameters (`query`), welcher schließlich mit dem Aufruf der Web Service-Funktion `findObjects` an das Repository übergeben wird. Das Ergebnis des Aufrufs wird anschließend in der Variable `$response` zwischengespeichert und nach einer Konvertierung der Ergebnisse zur Vereinheitlichung durch die Funktion `map`[19] an die aufrufende Stelle übermittelt.

Die hier gezeigte `simpleSearch`-Funktion unterscheidet sich nicht von den weiteren Funktionen zum Zugriff auf die Datenbestände (`moreResults` und `detailsForObject`). Jene sind nach dem selben Schema aufgebaut und im Anhang C.2 ab Seite 158ff nachzulesen. Sie unterscheiden sich von `simpleSearch` lediglich durch die übergebenen Parameter und die Aufrufe anderer Web Service-Funktionen.

6.2.4 Integration der Client-Anwendung in die Systemarchitektur

Die Client-Anwendung wurde dahingehend entwickelt, Plattformunabhängigkeit und Interoperabilität zu gewährleisten. Jene wurden aufgrund dessen durch die Verwendung standardisierter Protokolle und Sprachen zur Realisierung von Schnittstellen hergestellt (SOAP, WSDL und HTTP).

Die Verbindungen der Client-Anwendung mit den beteiligten Systemen sind aus folgendem Diagramm ersichtlich. In der Abbildung sind die Repositories *Pangaea* und *Fedora* angeschlossen. Mit dem rechtsstehenden, monochrom eingefärbten Repository soll deutlich gemacht werden, dass beliebige weitere Repositories optional und dynamisch in das System eingebunden werden können.

[18] In diesem Fall ist die WSDL-Beschreibung als Klassenvariable vorgegeben. Der Inhalt wird im Konfigurationsinclude `config.inc.php` mit folgendem Wert gesetzt: `http://web.awi-bremerhaven.de:8080/fedora/access/soap?wsdl`

[19] Funktion zur Vereinheitlichung der Ergebnisse. Stellt sicher, dass alle Ergebnisse nach dem selben Schema XML-kodiert (vgl. Listing 6.39) zurückgeliefert werden.

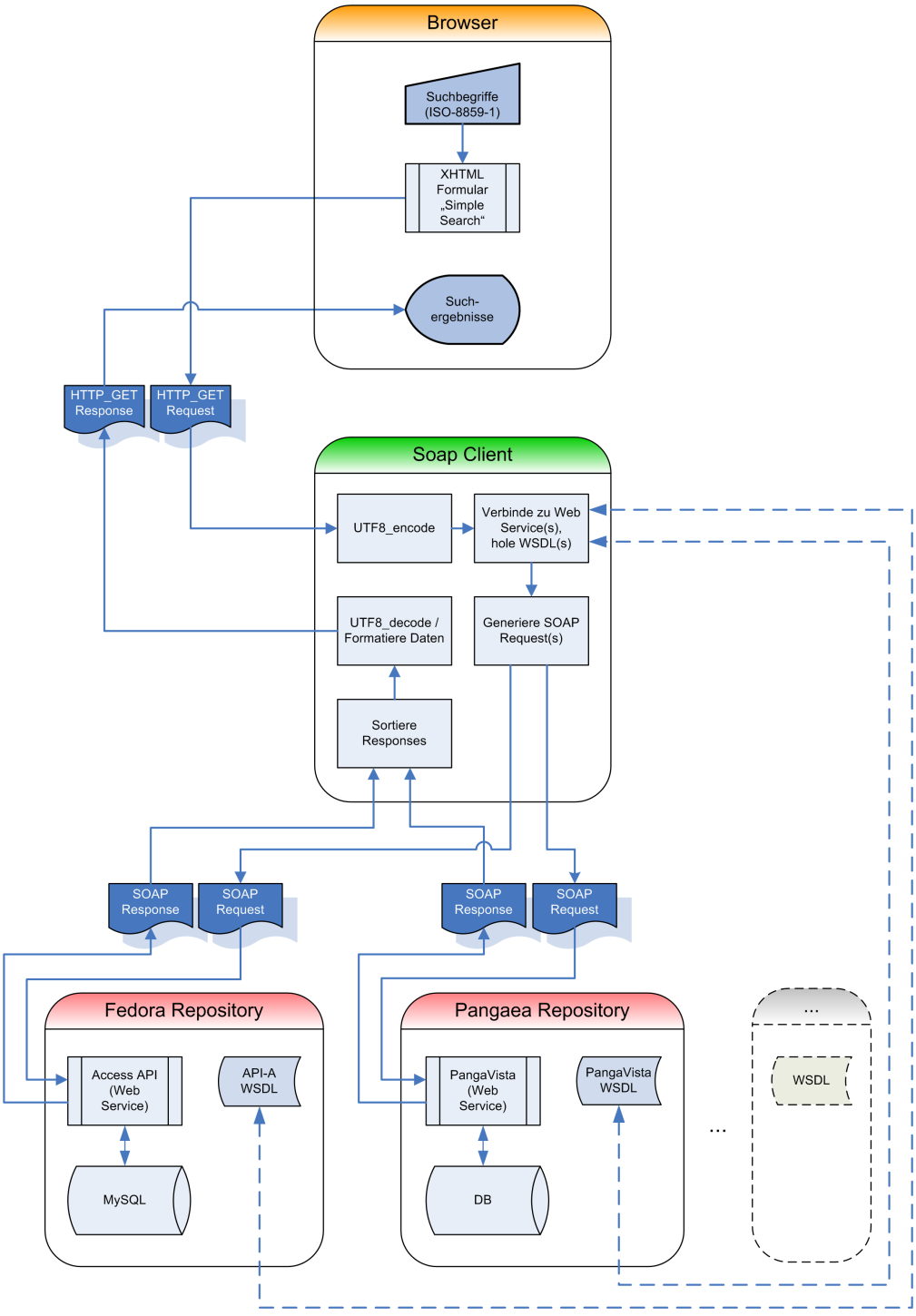


Abbildung 6.5: Architektur des Systems

6.2.5 Realisierung gleichzeitiger Suchanfragen

Um die gleichzeitige Abarbeitung von Anfragen an Repositories durchführen zu können, muss eine Multithreading-fähige Umgebung eingerichtet werden. Leider sieht PHP in der genutzten Version 5 nicht vor ein Multithreading von Grund auf zu ermöglichen, da die Ziele dieser Programmiersprache auf anderen Schwerpunkten liegen (für die nahe Zukunft ist es auch nicht angedacht diesen Umstand zu ändern). Nichtsdestotrotz ist es möglich in PHP eine Multithreading-fähige Anwendung zu erstellen. Dies ist allerdings durch einen nicht unerheblichen Aufwand möglich, denn die Programmbibliothek (`cURL`[20]), die hier dazu verwendet wird, ist noch teilweise als „experimentell“ einstufbar.

6.2.5.1 Methodik einer Suchanfrage

Das Skript, welches das Suchformular bereitstellt und die Suchanfragen entgegennimmt (`index.php`), übergibt dazu die Suchworte als String an ein weiteres Skript (`webServiceCaller.php`). Der Aufruf wird dabei als HTTP-GET-Anweisung mit Hilfe der *cURL*-Bibliothek umgeleitet. Die Bibliothek ermöglicht es, gleichzeitig mit mehreren Servern zu verbinden und über diverse Protokolle – hier HTTP – zu kommunizieren. Das Skript `webServiceCaller.php` wird somit bei einer Suchanfrage mehrfach ausgeführt und mit der Angabe von Parametern über die URL für den Zugriff auf ein Repository individuell konfiguriert.

Eine URL zum Aufruf eines Web Services ist wie folgt aufgebaut:

```
http://web.awi-bremerhaven.de/php/Helmholtz_WS/webServiceCaller.php?mode=simpleSearch&classid=0&
    terms=atlantic
```

Die Angabe `mode` legt die Art der Suche fest. Mögliche Modi sind „`simpleSearch`“ und „`moreResults`“. Mit `classid` wird der in der Konfigurationsdatei angegebene Web Service angesprochen. Mit der Angabe „`0`“ wird der erste Web Service angesprochen (mit „`1`“ der Zweite usw.). Das Argument `terms` übergibt letztendlich die Suchworte.

Die Antwort nach dem Aufruf der oben angegebenen URL wird XML-kodiert zurückgegeben. In diesem Fall wird nach „atlantic“ gesucht – die Ergebnisse stellen sich wie folgt dar:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <search>
3  <result>
4   <dc_title>Personal Homepage of Ismael Nunez-Riboni</dc_title>
5   <dc_creator>Ismael Nunez-Riboni</dc_creator>
6   <dc_date>2005</dc_date>
7   <dc_type>text, people</dc_type>
```

[20] Integriert in PHP, muss allerdings aktiviert werden, vgl Abschnitt 6.1.2.1 auf Seite 83. Siehe dazu auch `http://curl.haxx.se/libcurl/`.

```
 <dc_identifier>awi:inunez</dc_identifier>
</result>
<result>
 <dc_title>Cenozoic bottom current sedimentation in the Cape Basin, South Atlantic</dc_title>
 <dc_creator>Wildeboer Schut, E.</dc_creator>
 <dc_creator>Uenzelmann-Neben, G.</dc_creator>
 <dc_date>2005</dc_date>
 <dc_type>text, article</dc_type>
 <dc_identifier>awi:Eti2001a</dc_identifier>
</result>
<result>
 <dc_title>Evaluation of an eddy-permitting finite-element ocean model in the North Atlantic</
     dc_title>
 <dc_creator>Danilov, S.</dc_creator>
 <dc_creator>Kivman, G.</dc_creator>
 <dc_creator>Schröter, J.</dc_creator>
 <dc_date>2005</dc_date>
 <dc_type>text, article</dc_type>
 <dc_identifier>awi:Dan2004a</dc_identifier>
</result>
<result>
 <dc_title>Modeling the Speciation and Biogeochemistry of Iron at the Bermuda Atlantic Timeseries
     Study Site</dc_title>
 <dc_creator>Weber, L.</dc_creator>
 <dc_creator>Voelker, C.</dc_creator>
 <dc_creator>Schartau, M.</dc_creator>
 <dc_creator>Wolf-Gladrow, D. A.</dc_creator>
 <dc_date>2005</dc_date>
 <dc_type>text, article</dc_type>
 <dc_identifier>awi:Web2005a</dc_identifier>
</result>
<result>
 <dc_title>Simulated changes in vegetation distribution, land carbon storage, and atmospheric CO2
     in response to a collapse of the North Atlantic thermohaline circulation</dc_title>
 <dc_creator>Köhler, P.</dc_creator>
 <dc_creator>Joos, F.</dc_creator>
 <dc_creator>Gerber, S.</dc_creator>
 <dc_creator>Knutti, R.</dc_creator>
 <dc_date>2005</dc_date>
 <dc_type>event, poster</dc_type>
 <dc_identifier>awi:Khl2005c</dc_identifier>
</result>
<technicalinfo>
 <ws>Fedora at AWI</ws>
 <wsid>0</wsid>
 <session>d557ed2e9b42f0dcf355054db67e1c3f</session>
 <offset>0</offset>
 <responsetime>0.121s</responsetime>
</technicalinfo>
</search>
```

Quelltext 6.39: Ergebnis einer Suchanfrage durch webServiceCaller.php – Suchwort „atlantic“

Die Ausgabe mit `mode=simpleSearch` und `classid=0` reflektiert die Ausgabe der *simpleSearch*-Methode des Web Service mit der Identifikationsnummer „0“ (hier: Fedora). Die Ansicht der Eigenschaften ist auf die Dublin-Core-Felder *title*, *creator*, *date*, *type* und *identifier* beschränkt, da davon ausgegangen wird, dass der Benutzer mit seiner Suchanfrage nicht alle Details der zurückgelieferten Objekte angezeigt bekommen möchte. Aus diesem Grund werden nur Objekt-Vorschauen zurückgeliefert. Detailliertere Informationen zu einem Objekt können allerdings mit der Funktion *detailsForObjekt* abgerufen werden.

Am Ende einer jeden Suchanfrage werden technische Informationen zur Abarbeitung des Requests angehängt. Diese enthalten Informationen zur Identifikation einer Antwort sowie zur späteren Generierung weiterer Anfragen durch die Angabe von *Session-ID*[21] und *Offset*.

6.2.5.2 Erzeugen einer Multithreading-Umgebung

Die cURL-Bibliothek führt nun je Web Service einen solchen HTTP-GET-Befehl parallel aus und gibt die sich daraus ergebenden individuellen XML-Dokumente zur Anzeige zurück. Durch die Umleitung über HTTP-GET mittels `cURL` wird somit eine Umgebung emuliert, in der mehrere Funktionsaufrufe gleichzeitig durchgeführt werden können. Da die Aufrufe direkt an den selben Web Server gerichtet sind, muss dieser dazu in der Lage sein, gleichzeitige Anfrage zu bearbeiten. Der Apache HTTP Server ist durch die Integration des MPM-Moduls dazu bestens vorbereitet.

Der Ablauf der Anfrage ist im Folgenden detailliert aufgeführt:

[21] Fehlt die Angabe einer Session-ID, existieren keine weiteren Elemente im Repository die der Suchanfrage entsprechen.

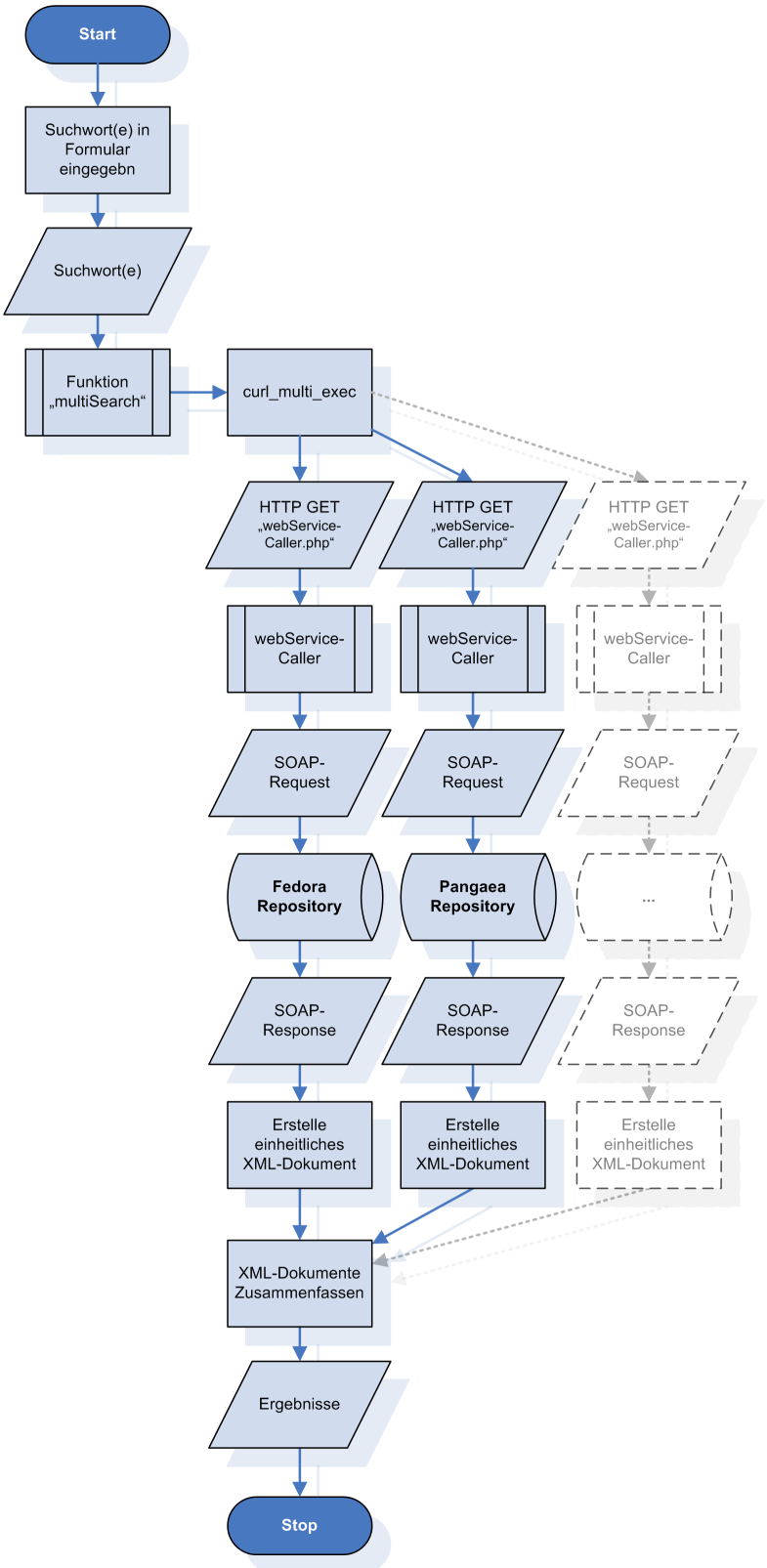


Abbildung 6.6: Programmablaufplan Publikationssuche

6.2.6 Grenzen des Prototyps

Wie bereits in der Zielformulierung zu Beginn der Dokumentation auf Seite 3 erwähnt, soll im Rahmen dieser Diplomarbeit ein Prototyp entwickelt werden, der zur Demonstration der Verwendbarkeit von SOAP-API's von Repositories dienen soll. Natürlich ist die Anwendung auch für eine zukünftige produktive Nutzung angedacht, die Entwicklung eines Produktes mit vollem Funktionsumfang ist aber nicht Inhalt der Aufgabe.

Aus diesem Grund soll an dieser Stelle ein Vergleich mit einem möglichen, wünschenswerten, realistischen Endprodukt angestellt werden, der die noch zu erarbeitenden Eigenschaften des Prototyps aufzeigt, aber auch eine Vision auf das Endprodukt gibt, wie es vielleicht irgendwann einmal Verwendung am AWI oder an anderen Einrichtungen finden wird.

	Prototyp	**Endprodukt**	**Bemerkung**
Performanz	gut bis sehr gut	sehr gut	Optimierungen an den Quelltexten könnten Leistungssteigerungen bringen.
Wartbarkeit	befriedigend	sehr gut	Das Emulieren einer Multithreading-Umgebung hat die Anwendung sehr undurchsichtig werden lassen. Eine Umorganisation der Quelltexte könnte eine erhöhte Wartbarkeit erzielen.
Erweiterbarkeit (Hinzufügen weiterer Repositories)	sehr gut	sehr gut	Die Erweiterbarkeit ist bereits in vollem Umfang integriert. Per Konfiguration können Repositories optional hinzugefügt und entfernt werden. Allerdings sind PHP-Kenntnisse dazu erforderlich.
Fehlertoleranz	gegeben, aber perfektionierbar	vollständig gegeben	Für das Erreichen einer vollständigen Fehlertoleranz sind viele Tests nötig, die den Rahmen dieser Diplomarbeit sprengen – für den Produktivbetrieb allerdings unbedingt notwendig.
Einführen eines Sessionmanagements	nicht implementiert	implementiert	Wünschenswert ist die Integration von Sessions. Diese würden die Kommunikation zwischen den Skripten erleichtern, was die Wartbarkeit ebenfalls steigern würde.
Integration einer Datenbank	nicht implementiert	nicht implementiert	Die Verwendung einer Datenbank würde die Einfachheit der Anwendung zerstören: Denn ein Ziel ist die zu erreichende Plattformunabhängigkeit der Anwendung.
Erweiterte Suchmöglichkeiten	nicht implementiert	implementiert	Die Grundlage für eine erweiterte Suche ist aber bereits geschaffen. Es mangelt momentan an der Unterstützung seitens Fedora.

Tabelle 6.3: Produktvergleich

Kapitel 7

Dokumentation der Anwendung

Zur Dokumentation sind im Folgenden die erarbeiteten Seitenelemente aufgeführt, wie sie in der Client-Anwendung Verwendung finden. Sie ergänzen das grundlegende Seitenkonzept aus der Prototypenentwicklung aus Abschnitt 5.2.3 und komplettieren somit die Benutzerschnittstelle. Nach der Element-Beschreibung knüpft eine beispielhafte Sitzung an, welche die Funktionsweise der Anwendung anhand der fertigen Benutzeroberfläche demonstriert.

7.1 Elemente der Benutzeroberfläche

7.1.1 Banner

Der folgende Banner-Entwurf lässt sich auf jeder angezeigten Seite der Anwendung wiederfinden. Er dient zur Identifikation und zeigt gleichzeitig die Zugehörigkeit zu Open Access und Helmholtz-Gemeinschaft. Aus der Restriktion der maximalen Seitenbreite durch die festgelegte minimale Bildschirmauflösung ergibt sich eine Bannerbreite von 700 Pixel:

Abbildung 7.1: Banner

7.1.2 Navigation

Für das Design der Navigationsleiste wurden Text-Links verwendet. Die Entscheidung dazu resultiert aus der Forderung danach, die Anwendung so einfach wie möglich durch die entstandenen selbst-beschreibenden Titel zu halten:

Abbildung 7.2: Navigation

Abbildung 7.3: Hover Effekt

Abbildung 7.4: Visited Link

Da die Navigationsleiste durch dessen Verknüpfungseigenschaften vom übrigen Text abgehoben werden soll, ist zum Einen eine differierende Linkfarbe gewählt worden und zum Anderen ein „Hover“-Effekt eingesetzt worden. Der „Hover“-Effekt wird sichtbar, wenn der Mauszeiger über eine Verknüpfung bewegt wird. Diese wird durch die Veränderung der Hintergrundfarbe auf Weiß dann als aktivierbarer Link gekennzeichnet.

Zur Verbesserung der Übersicht, wurden verschiedene Farbtöne zur Kennzeichnung bereits gesehener Web Seiten eingesetzt. Diese werden im Gegensatz zu nichtbesuchten Web Seiten in einem Grauton dargestellt.

7.1.3 Suchformular

Das Suchformular wird von einem Text eingeleitet, der die durchsuchbaren Inhalte in einem kurzen Satz erläutert. Die Eingabemöglichkeit ist auf ein Feld beschränkt, in welches Suchworte eingegeben werden können. Ein Hinweis auf die Platzierung der Suchworte durch den Ausdruck „insert search terms here“ zeigt dem Benutzer, welche Inhalte er in das Formular eingeben muss[1]. Zur Auswahl der zu durchsuchenden Repositories sind an- und abwählbare Checkboxen eingefügt worden, welche per Mausklick bedient werden können. Eine Anfrage wird gestartet, sobald der Benutzer die Schaltfläche „Search“ aktiviert.

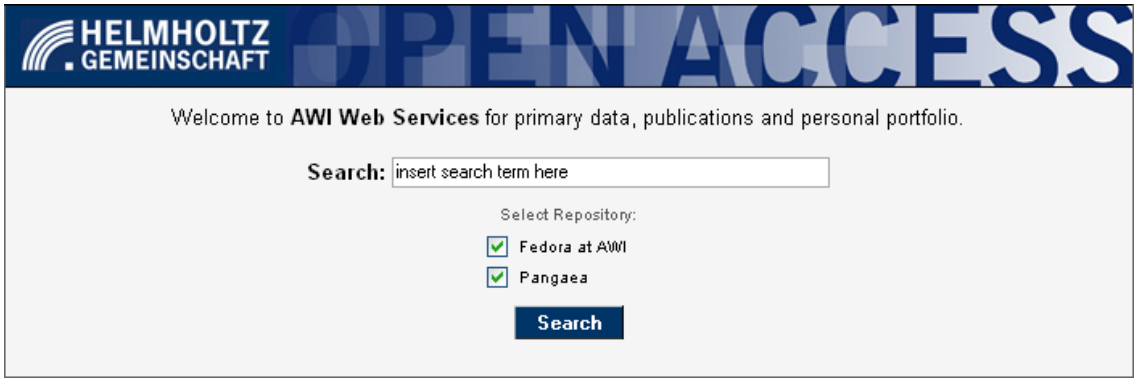


Abbildung 7.5: Suchformular

[1]Der Hinweistext wurde durch Vorschrift der BITV-Verordnung, Priorität III eingefügt

7.1.4 Ergebnisanzeige

Der Suchanfrage folgt die Anzeige der Suchergebnisse – vorausgesetzt, die Suche war erfolgreich. In der Ergebnisansicht werden die Objekte, welche der Suchanfrage entsprechen, ausschnittartig angezeigt. Das heißt, es werden nicht alle Details eines Objektes ausgegeben. Zur Wahrung der Übersicht beschränkt sich die Ausgabe in einer Liste mit *Titel*, *Format*, *Datum* und *Autor*.

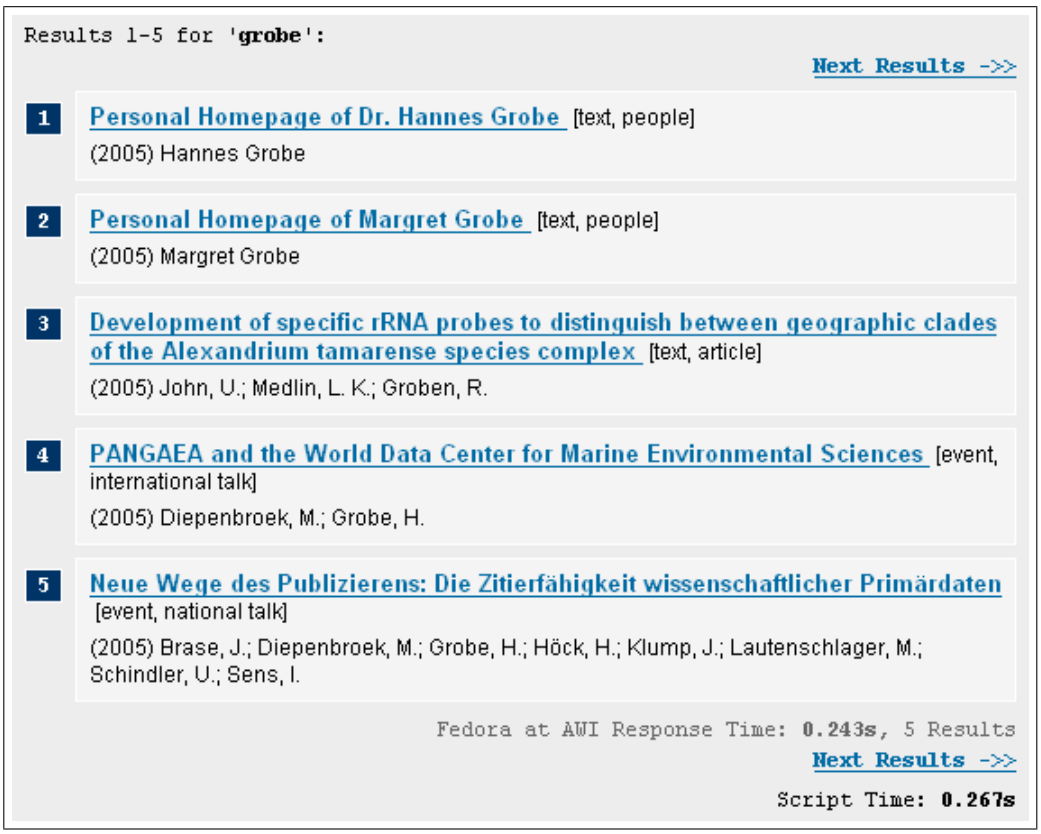


Abbildung 7.6: Ergebnisse

Sind detailliertere Informationen zu einem Objekt erforderlich, ist die Detailansicht eines Objektes mit dem Titel verknüpft. Ein Mausklick auf diesen öffnet die folgende Ansicht, welche alle Felder eines Objektes enthält. Enthält ein zurückgeliefertes Feld eine Link, wird diese automatisch mit dem Ziel verknüpft und ist direkt in dieser Ansicht aufrufbar (bspw. Link zur Quelle):

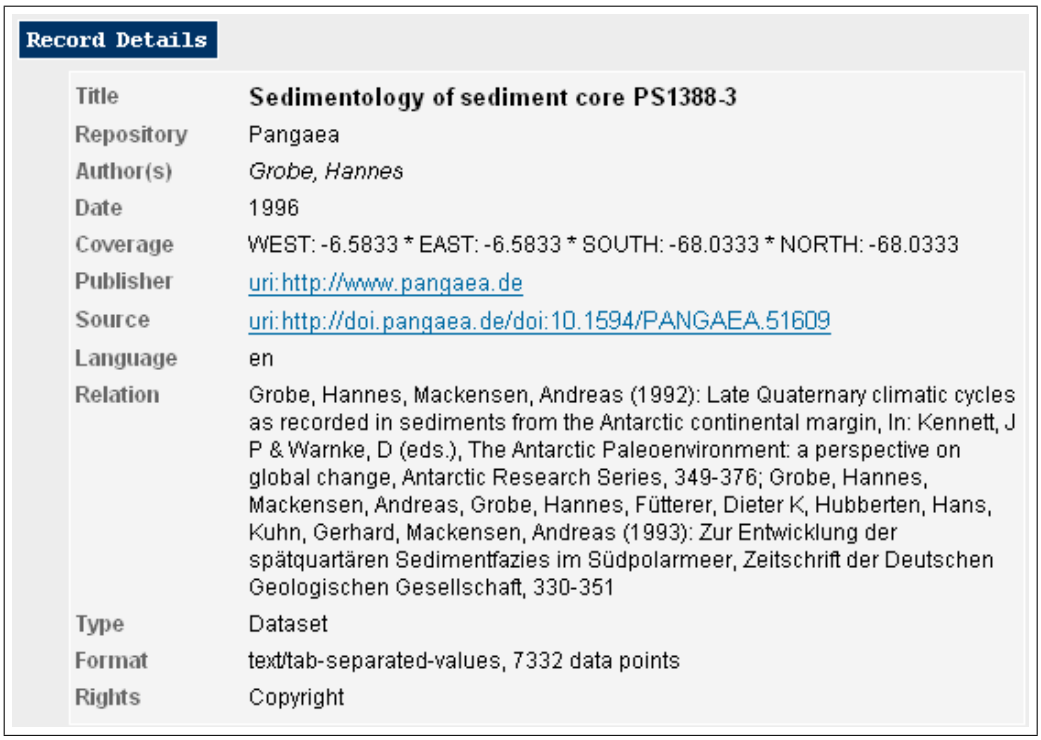


Abbildung 7.7: Detailansicht

7.1.5 Validierung der Benutzeroberfläche

Die gesamte Benutzeroberfläche wurde daraufhin optimiert, die Richtlinien der BITV-Verordnung in der Priorität III einzuhalten[2]. Diese ist die höchste der insgesamt drei Prioritätsstufen und steht für eine erweiterte Zugreifbarkeit. Zur Validierung wurden zur Erreichung dieser Stufe drei unterschiedliche Werkzeuge zur Überprüfung eingesetzt: *WebXACT* der Watchfire Corporation[3], *Cynthia Says* von HiSoftware[4] und *Accessibility Valet* (WebThing[5]).

[2] vgl. Abschnitt 5.1.5.2, Seite 67
[3] s. `http://webxact.watchfire.com/`
[4] s. `http://www.contentquality.com/`
[5] s `http://valet.webthing.com`

7.2 Beispielhafte Sitzung

Die Demonstration der Funktionalitäten soll anhand der folgenden, beispielhaften Sitzung veranschaulicht werden. Dazu sind Screenshots der einzelnen Ereignisse abgedruckt, welche die Benutzeroberfläche der Anwendung im Fenster eines Web Browser anzeigen.

7.2.1 Startseite

Ausgangsposition der Anwendung ist die Startseite, welche durch das Aufrufen des Skripts `index.php` generiert wird. Inhalt dieser Seite ist das Suchformular und die Navigationsleiste. Am unteren Ende auf dieser (und allen weiteren Seiten) sind zusätzliche Informationen zur Qualität des Quelltextes in Form von kleinen, unauffälligen Bildern enthalten, welche mit entsprechenden Validierungswerkzeugen verknüpft sind.

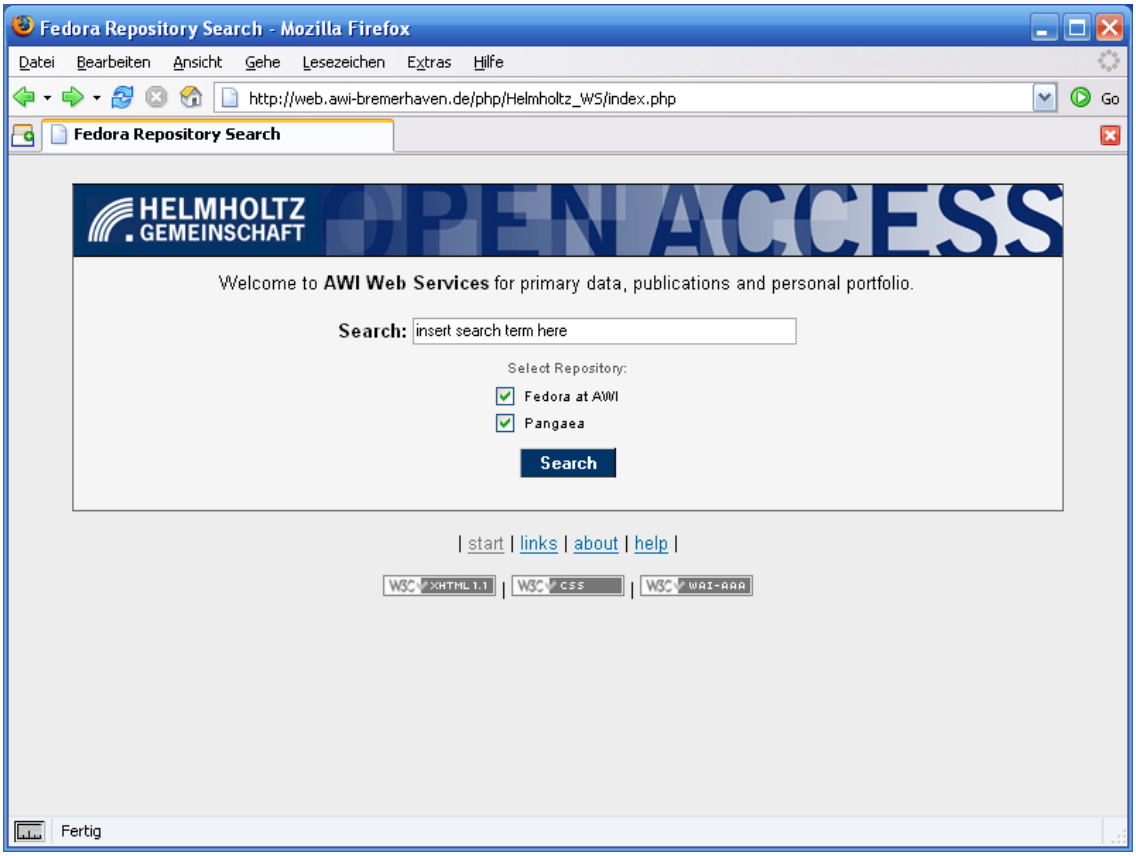


Abbildung 7.8: Startseite

7.2.2 Suche nach Publikationen

Über das Suchformular der Startseite werden Suchanfragen an Repositories gestellt. Um den Ablauf dieser Anfrage zu erläutern zeigen folgende Abbildungen den Ablauf der durchzuführenden Aktionen bis zur Anzeige von Ergebnissen in der Detailansicht.

7.2.2.1 Auswahl der Repositories

Nach der Angabe eines Suchwortes (in diesem Beispiel wurde „pfeiffenberger“ angegeben) können die Repositories ausgewählt werden, an jene die Suchanfrage gesendet werden soll. Dazu sind die angezeigten Repositories durch Kontrollkästchen an- und abwählbar. Standardmäßig sind alle Repositories aktiviert. Im Folgenden Beispiel wird nur ein Repository ausgewählt, die Option „Pangaea“ wird durch einen Mausklick deaktiviert:

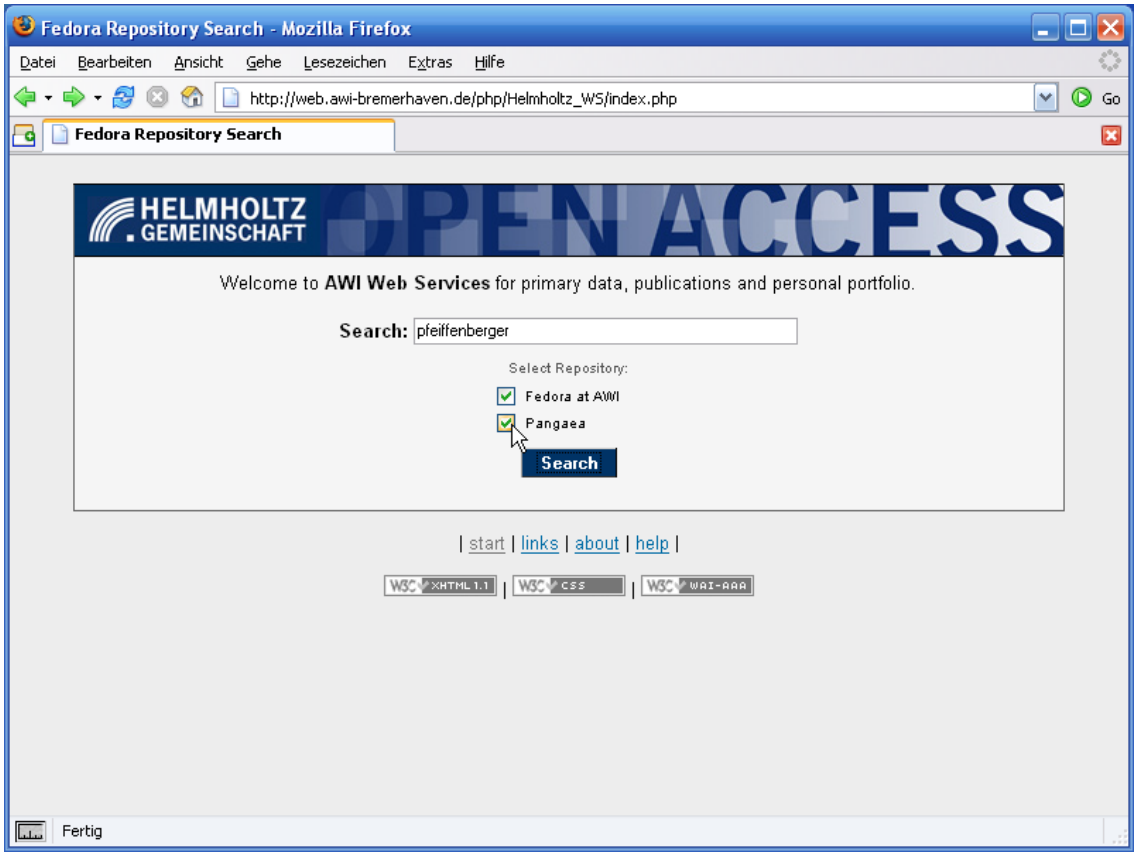


Abbildung 7.9: Repository Auswahl

7.2.2.2 Anzeige von Suchergebnissen

Alle Suchparameter sind nun getätigt worden. Die folgende Abbildung zeigt die Ergebnisse der oben spezifizierten Suchanfrage nachdem die Anfrage mit einem Mausklick auf die Schaltfläche „Search“ abgesendet und von den angeschlossenen Repositories beantwortet wurde.

Abbildung 7.10: Anzeige der Ergebnisse

7.2.2.3 Anzeige weiterer Ergebnisse

Falls die Anzahl der zurückgelieferten Ergebnisse die in der Programmkonfiguration festgelegte maximale Listenlänge überschreitet, werden Links zur Anzeige weiterer Ergebnisse angezeigt. Ein Klick auf einen Link dieser Art macht ein vorwärtsgerichtetes, sowie rückwärtiges Navigieren durch die gesamte Ergebnisliste möglich:

Abbildung 7.11: Weitere Ergebnisse anzeigen

7.2.2.4 Detailansicht eines Ergebnisses

Da in der Ergebnisliste lediglich verkürzte Informationen angezeigt werden, ist es durch das Anklicken des Titels eines Eintrags möglich, eine Detailansicht eines Objektes aufzurufen. Diese Ansicht enthält alle Felder eines Objektes:

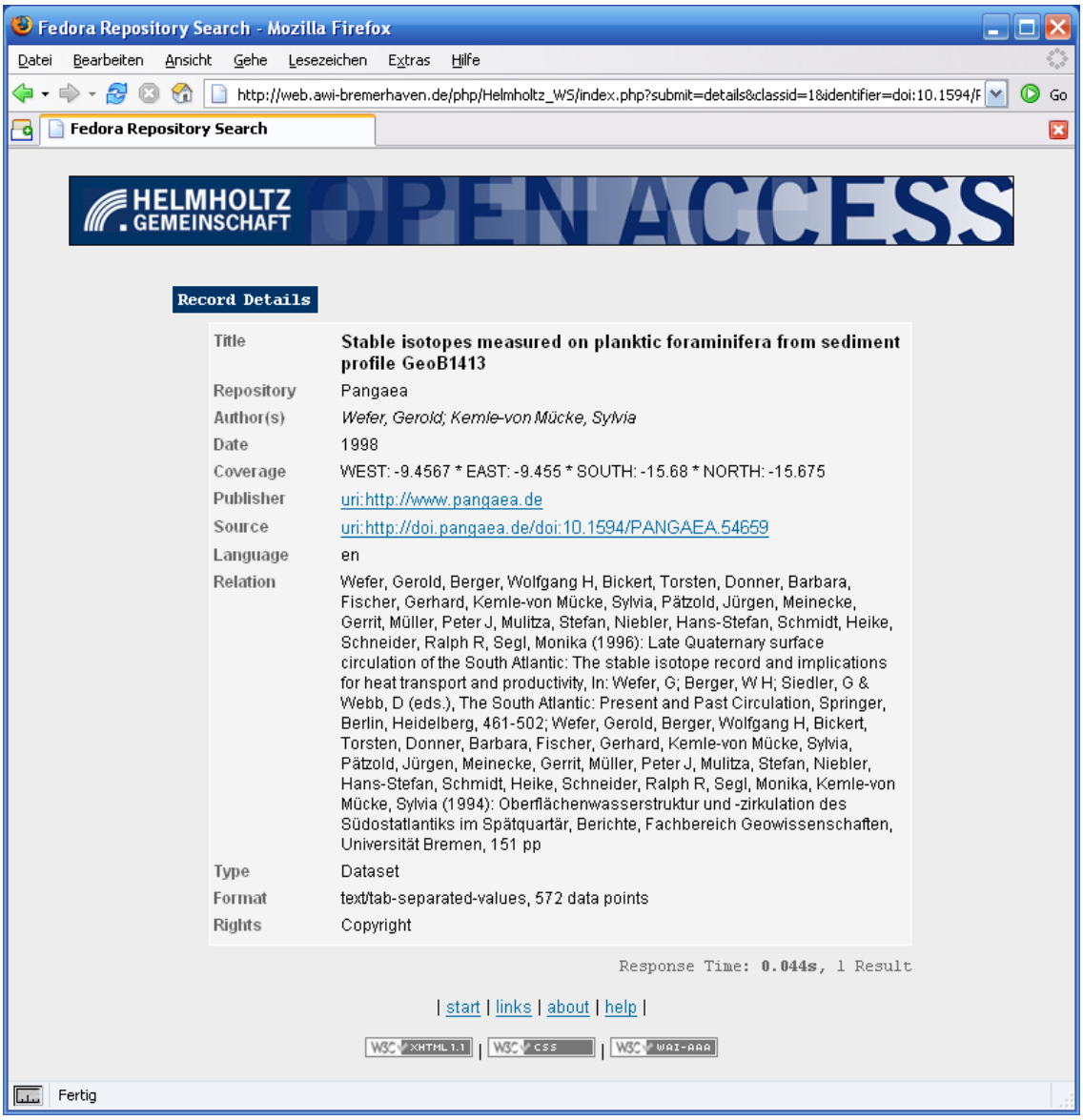


Abbildung 7.12: Detailansicht eines Ergebnisses

7.2.3 Fehlerbehandlung

Das Konzept zur Entwicklung des Prototyps sieht eine besonders filigrane Fehlerbehandlung vor. Im Folgenden sind zwei absichtlich herbeigeführte, mögliche Ursachen für falsche Eingaben und dessen anschließende Behandlung abgebildet.

7.2.3.1 Leere Eingabe

Das Eingeben eines leeren Suchwortes wird keine sinnvollen Ergebnisse zurückliefern. Aus diesem Grund ist die Eingabe von Leer-Strings durch eine Überprüfung im Skript verhindert worden. Statt eine Anfrage an die Repositories zu senden, wird in diesem Fall eine Fehlermeldung mit dem Inhalt: „Please insert a valid search term!“ ausgegeben. Die Farbe Rot signalisiert dem Benutzer, dass seine Anfrage fehlerhaft war und aufgrund dessen nicht bearbeitet werden konnte:

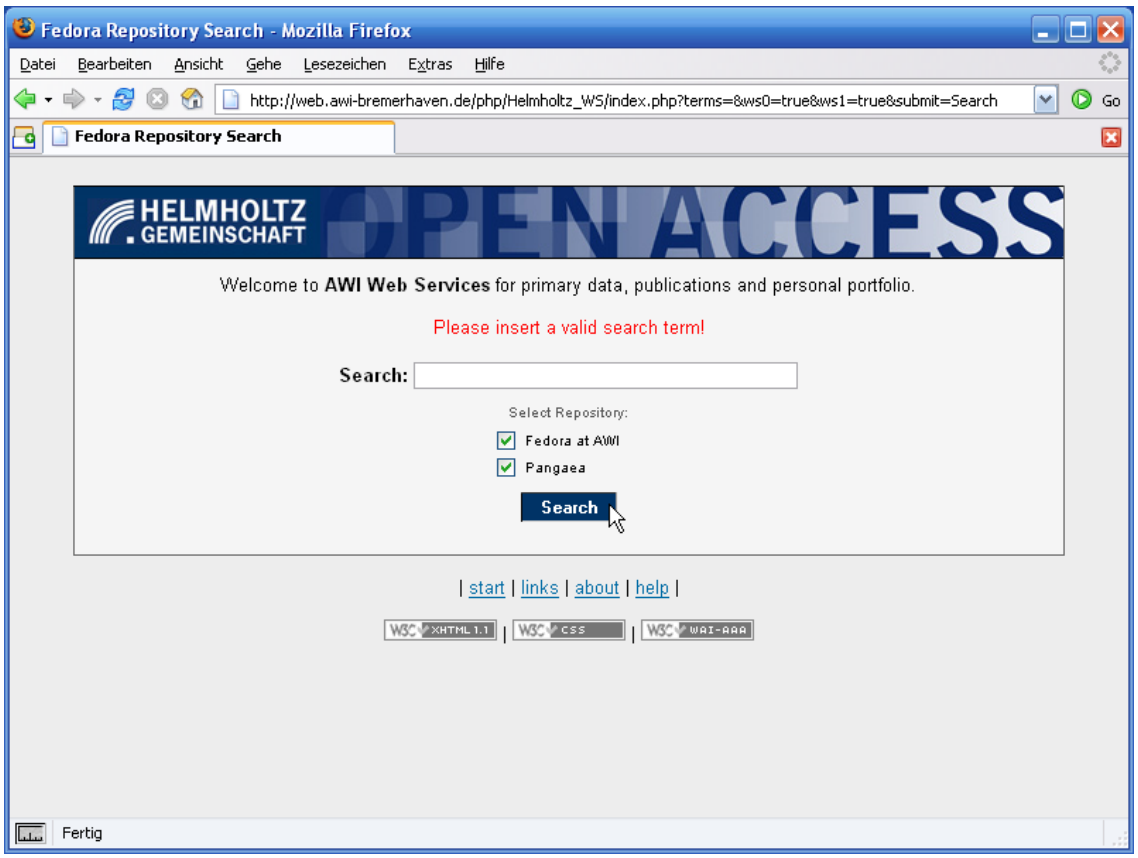


Abbildung 7.13: Leere Eingabe

7.2.3.2 Keine Ergebnisse gefunden

Der zweite Fehlerfall tritt ein, wenn zu einem Suchwort keine Ergebnisse gefunden werden konnten. Zur Mitteilung dessen an den Benutzer wird eine Meldung mit dem Text „No entries found for query '`Suchwort`'...“ ausgegeben, welche ebenfalls durch die Farbe Rot besonders gekennzeichnet wird.

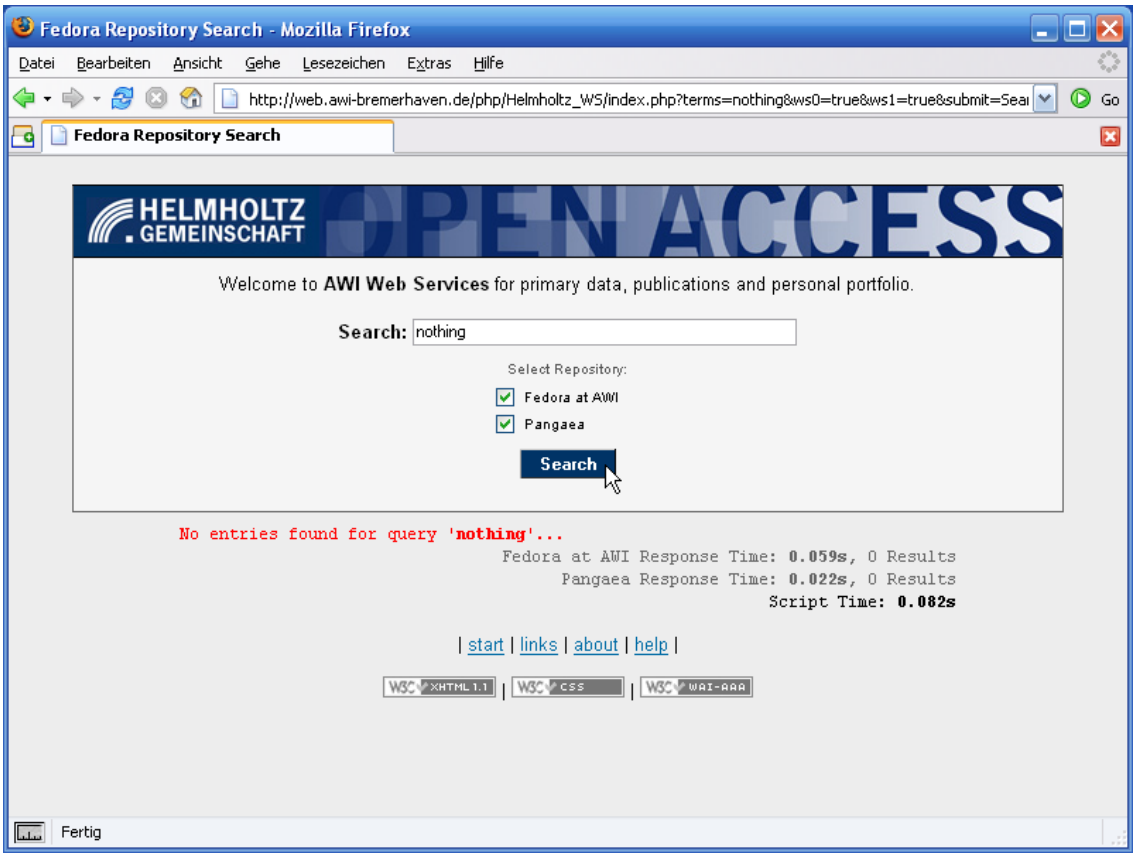


Abbildung 7.14: Keine Ergebnisse gefunden

7.2.4 Informationen und Hilfe

Weiterer Bestandteil der Anwendung sind Webseiten zur Darstellung von Informationen über die Anwendung, sowie Hilfestellungen zur Nutzung der Suchfunktion.

7.2.4.1 Linkseite

Zur Information über die mit dieser Anwendung verknüpften Repositories, ist eine Link-Seite entworfen worden. Diese enthält zu jedem Repository eine kurze Beschreibung und Verknüpfung zur Internetpräsenz.

Abbildung 7.15: Links

7.2.4.2 Anzeige der Verantwortlichen

Die Seite zur Anzeige der Verantwortlichen ist ähnlich aufgebaut. Sie enthält die Namen der Entwickler und Verantwortlichen der Anwendung. Zusätzlich sind Email-Adressen, Telefon- und Faxnummern zur Kontaktaufnahme enthalten.

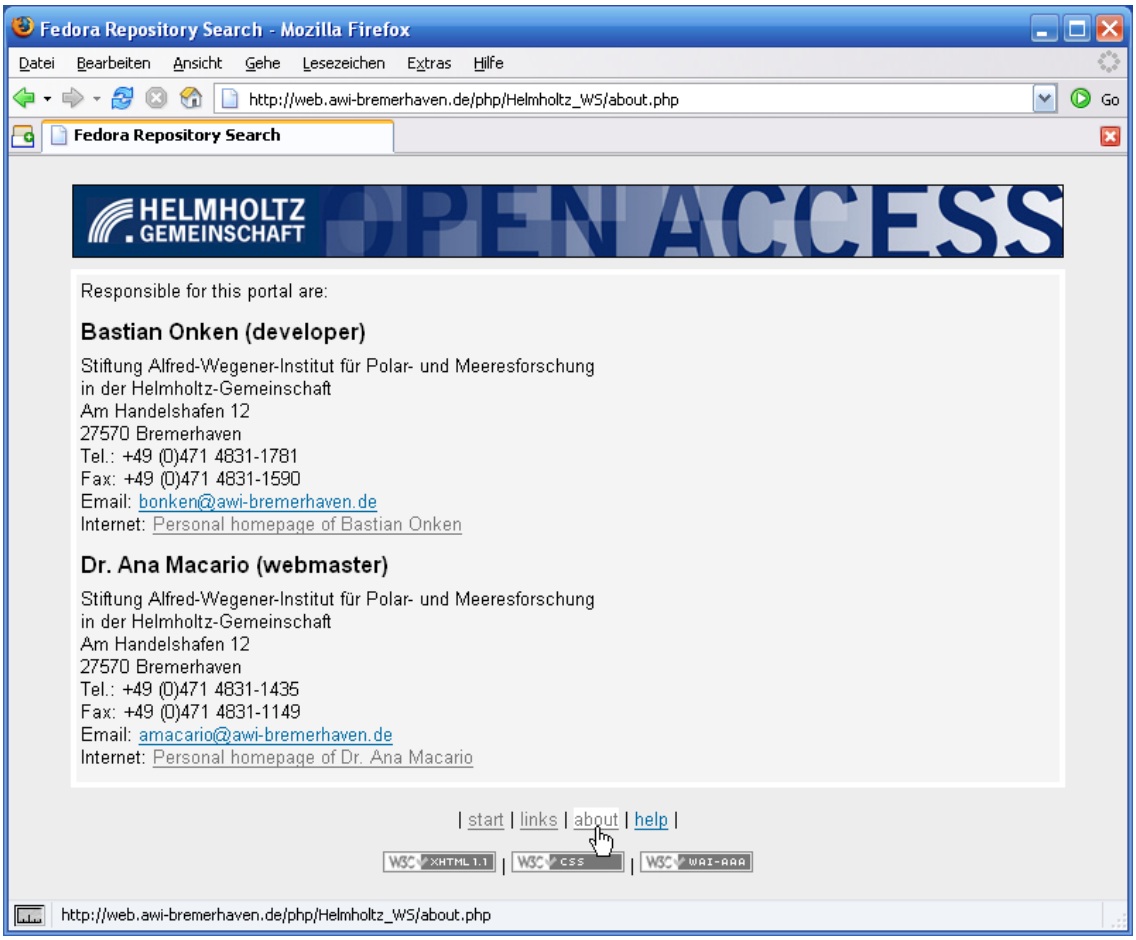


Abbildung 7.16: Verantwortliche

7.2.4.3 Anzeige der Hilfe

Das Hilfekonzept ist in eine separate Webseite integriert worden. Die Navigationsleiste weist mit der Aufschrift „help“ darauf hin, dass an dieser Stelle Informationen zur Verfügung stehen, die Unterstützung bei der Bedienung der Anwendung geben können.

Die Hilfe ist in mehrere Unterpunkte untergliedert und wie alle verwendeten Texte der Anwendung in englischer Sprache verfasst worden:

Abbildung 7.17: Hilfe

Kapitel 8

Zusammenfassung und Ausblick

8.1 Repositories an wissenschaftlichen Einrichtungen

Es ist nicht unbekannt, dass eine hohe Anzahl anfallender Publikationen jeglicher Art einer hochintensiven Administration und Pflege bedarf. Gleichzeitig müssen die Publikationen insoweit verfügbar gemacht werden, dass ein Wiederauffinden mit geringem technischen und zeitlichen Aufwand möglich ist – idealerweise ist die Benutzerschnittstelle dabei ein plattformunabhängig agierender Web-Browser. Da wissenschaftlicher Output selten in verallgemeinerte Schemata passt, müssen zwangsläufig individuelle Objekttypen geformt werden, die den Bedürfnissen der Wissenschaft angepasst werden müssen. Eine einfache Archivierung in einer simplen Datenbankumgebung kann dabei schnell an Grenzen stoßen.

Repositories hingegen zeichnen sich dadurch aus, genau diese erforderlichen Grundvoraussetzungen zu erfüllen und in einem Produkt zu vereinen. Wichtige Funktion sind hier die flexiblen, selbst definierbaren Datenstrukturen für Objekte, welche auf Langzeitarchivierung ausgelegt sind, und die Verwendbarkeit von lose gekoppelten Anwendungen zu nennen, die es ermöglichen, den Zugriff auf die Inhalte mit standardisierten Mechanismen effizient zu organisieren. Des Weiteren sind enthaltene Objekte durch ein Versionsmanagement in allen Zuständen ihres Lebenszyklus' verfügbar, was die Verfolgung von Arbeitsgängen möglich macht und zusätzlich vor möglichem Datenverlust absichert.

Ob ein Repository durch OAI-PMH oder Web Services realisiert wird, ist dabei eine wichtige, zu berücksichtigende Frage, die sich die Administration einer Datenhaltung für wissenschaftlichen Output stellen muss. Beide Lösungen bieten technisch einwandfreie Mittel zur Realisierung von Repositories, allerdings unterscheiden sich

beide Architekturen in verschiedener Hinsicht[1], so dass bei einer Implementierung genauestens abgewägt werden muss.

Im Rahmen der Aufgabenstellung und Zielsetzung dieser Diplomarbeit ist der Einsatz der Web Service-orientierten Technologie als geeignetere Lösung angesehen worden. Dies bedeutet nun nicht, dass die Web Service-Technologie grundlegend Vorteile zur Schaffung von Open Access in einer wissenschaftlichen Großforschungseinrichtung bietet. Zu bedenken ist, dass durch den Einsatz von Web Services weitere Arbeitspakete entstehen (Programmieren von Schnittstellen zu anderen Anwendungen, bzw. Entwurf von eigenständigen Clients), welche mit OAI-PMH nicht zwingend aufgekommen wären. Zwar bietet das Repository-System (Fedora), welches in dieser Arbeit zum Einsatz gekommen ist, bereits vorgefertigte, integrierte Client-Anwendungen, diese sind allerdings in ihren Funktionen noch relativ beschränkt und nicht für den Endbenutzer-Einsatz geeignet[2].

8.2 Ausblick

8.2.1 Web Service-basierte Repositories

Die Vorteile von Web Service-basierten Repositories werden früher oder später die klassischen Ansätze von reinen OAI-PMH-Implementierungen teilweise ersetzen oder zumindest ergänzen können, davon kann mit hoher Sicherheit ausgegangen werden. Leider fehlt zur Zeit allerdings eine adäquate Standardisierung zur Implementierung von Institutionellen Repositories mittels Web Services, welche den Zugriff auf eine Datenhaltung mit vorgegebenen Funktionen festlegen könnte. Es wäre durchaus möglich eine Standardisierung global zu fixieren ohne diese an einen bestimmten Wissenszweig zu binden.

Trotz der Möglichkeit von individuellen Implementierungen Web Service-orientierter Schnittstellen liegen die entscheidenden Vorteile für Institutionelle Repositories auf Seiten der Web Services (siehe dazu Abschnitt 3.2, Seite 43ff). Das Indiz, dass die Urväter von OAI-PMH mit der Entwicklung von Fedora den Weg für Web Services-basierte Repository-Systeme beschritten haben, lässt ebenfalls erahnen, mit der Verwendung Web Service-basierter Implementierungen auf der zukunftsorientierten Seite zu stehen.

In der nahen Zukunft wird es verstärkt Web Service-orientierte Lösungen für Repositories geben – Fedora ist zur Zeit noch einer der Pioniere auf diesem Gebiet[3]. Die Problematik ist momentan, dass Web Services erst schleppend Einzug in die

[1] vgl. 3.2

[2] Zu berücksichtigen ist allerdings, dass das Fedora Repository-System noch in der Entwicklungsphase ist: Benutzerfreundliche Benutzeroberflächen werden voraussichtlich in zukünftige Versionen integriert.

[3] vgl. [BOA04, Seiten 11,17]

Unternehmenspraxis nehmen, dieses resultiert zu einem großen Anteil aus dem hohen Abstraktionsniveau von Web Services. Es ist allerdings nur noch eine Frage der Zeit, bis entsprechende Standardisierungen für Web Service-basierte Datenzugriffe vorliegen, die eine reibungslose Integration gewährleisten. Dies wird besonders die breite Akzeptanz Web Service-basierter Repositories fördern.

8.2.2 SOAP-Client

Um aufzuzeigen, welche Möglichkeiten eine Weiterentwicklung der im Rahmen dieser Diplomarbeit entwickelten Anwendung bieten könnte, soll aufgrund dessen im Folgenden auf optionale Erweiterungen eingegangen werden. Die Client-Anwendung ist bislang hervorragend zur Recherche institutsübergreifender Repositories einsetzbar. Allerdings lässt sich die Funktionalität durchaus erweitern:

- Das Problem der Datenverfügbarkeit, auf welches in Abschnitt 3.2.2.4 eingegangen wurde, könnte mit einer Fallback-Lösung umgangen werden. Ein im Hintergrund arbeitender OAI-PMH-Harvester könnte bei einem eventuellen Ausfall eines angeschlossenen Repository vorher gespeicherte Inhalte an den Nutzer ausliefern. Die dadurch ersetzten Daten sind zwar von einem älteren Datum, würden aber immerhin einen Datenbestand ausliefern.

- Für die nahe Zukunft ist eine Nachfolgeversion des Fedora-Repository in Planung. In dieser sollen Funktionen zur Workflowsteuerung den Zugriff auf Objekte der Datenhaltung mit globalen Authentifizierungsmaßnahmen organisieren.

  Das bereits bestehende Publikationssystem könnte durch die Integration von Workflows vollwertig durch Fedora ersetzt werden – die Client-Anwendung könnte in diesem Zuge mit erweiterten Funktionen versehen werden, welche den Lesezugriff von Objekten um ein Management zum Schreibzugriff ergänzt.

  Dabei könnte die erforderliche Authentifizierung von Fedora übernommen werden, damit Veröffentlichungen wissenschaftlicher Publikationen durch die Benutzer selbst vorgenommen werden können. Da die Web Service-Technologie vorgesehen ist in eine schon vorhandene Umgebung integriert zu werden, ist die Übernahme der bestehenden Formulare unproblematisch.

Grundsätzlich sagen lässt sich, dass das Potential der Client-Anwendung noch längst nicht ausgereizt ist. Da die Möglichkeiten zur Gestaltung von lose gekoppelten Anwendungen nahezu grenzenlos sind, kann jede erforderliche Funktionalität optional in die Systemarchitektur implementiert werden.

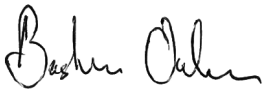
Kapitel 9

Persönliches Resümee

Die Arbeit der letzten Wochen stellt sich für mich als eine besondere Erfahrung in meiner Laufbahn als Student dar. Es gab nur wenige vergleichbare Arbeiten während der vergangenen Studienzeit, die zwar methodisch an die Arbeitsweise der Diplomarbeit heranreichten, nicht aber den Umfang dieser hiermit abgeschlossenen Erarbeitung umfassten. Die intensive Beschäftigung mit dem Thema *Open Access* und dessen Realisierung am Alfred-Wegener-Institut hat mir zahlreiche, einschlägige Kenntnisse vermittelt und bestehendes Wissen vertieft, welches mich für die anstehende berufliche Zukunft solide vorbereitet. Besonders sind die Themengebiete, die das Management von digitalen Inhalten betreffen und dessen praktische Umsetzung und Systemintegration in einem Unternehmen dieser Größe mit seinen weitreichenden, zu berücksichtigenden Organisationsstrukturen vermittelt worden. Zusätzlich habe ich erlernt, welches Potential hinter der Web Service-Technologie steht und wie sie produktiv eingesetzt werden kann.

Während der Erarbeitung der Ergebnisse konnte ich besonders die im Rahmen des Hochschulprojektes erworbenen Kenntnisse anbringen. Das zwei-semestrige Projekt beschäftigte sich mit einem Web Service-basierenden Online-Buchungssystem, welches von insgesamt 12 Studenten zu einem hervorragenden Abschluss gebracht wurde. Meine gesammelten Erfahrungen in diesem Projekt auf dem Gebiet der Web Services und als studentischer Projektleiter waren während meiner Arbeitszeit am Alfred-Wegener-Institut besonders hilfreich.

Abschließend möchte ich sagen, dass mir das Erarbeiten der Diplomarbeit trotz der vielen damit verbundenen Anstrengungen in positiver Erinnerung bleiben wird. Es würde mich freuen, wenn die Ergebnisse dieser Arbeit in der Zukunft weitere Verwendung finden würde – sei es am Alfred-Wegener-Institut oder an anderer Stelle.



Stotel, den 30. September 2005

Abkürzungsverzeichnis

API	Application Programming Interface
AWI	Alfred-Wegener-Institut für Polar- und Meeresforschung
BITV	Barrierefreie Informationstechnik-Verordnung
BMI	Bundesministerium des Innern
BOAI	Budapest Open Access Initiative
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CSV	Character Separated Values
DAML	DARPA Agent Markup Language
DC	Dublin Core
DCMI	Dublin Core Metadata Initiative
DCOM	Distributed Component Object Model
DFG	Deutsche Forschungsgemeinschaft
DMZ	Demilitarized Zone
DOI	Digital Object Identifier
DTD	Document Type Definition
EDV	Elektronische Datenverarbeitung
ePIC	Electronic Publication Information Center
EPK	Ereignisgesteuerte Prozesskette
FAQ	Frequently Asked Questions
FOXML	Fedora Object XML
FQS	Forum Qualitative Sozialforschung
FTP	File Transfer Protocol
GB	Gigabyte
GKSS	GKSS-Forschungszentrum Geesthacht GmbH
GUI	Graphical User Interface
HGF	Herrmann von Helmholtz-Gemeinschaft Deutscher Forschungszentren
HTML	Hypertext Markup Language

HTTP	HyperText Transfer Protocol
ID	Identifikationsnummer
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
JDBC	Java Database Connectivity
JDK	Java Development Kit
kB	Kilobyte
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MB	Megabyte
MHz	Megahertz
MPM	Multiprocessing Module
MS	Microsoft
NCSA	National Center for Supercomputing Applications
OA	Open Archives
OAI	Open Archives Initiative
OAI-PMH	OAI Protocol for Metadata Harvesting
OMG	Object Management Group
PC	Personal Computer
PDA	Personal Digital Assistant
PDF	Portable Document Format
PEAR	PHP Extension and Application Repository
PHP	Hypertext Preprocessor
PID	Persistent Identifier
PKP	Public Knowledge Project
PNG	Portable Network Graphics
RAM	Random Access Memory
RDF	Resource Description Framework
RMI	Remote Method Invocation
SAML	Security Assertion Markup Language
SGML	Standard Generalized Markup Language
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SP	Service Pack
SQL	Structured Query Language
T1	Trunk 1
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UTF-8	8-bit Unicode Transformation Format
W3C	World Wide Web Consortium

WAN	Wide Area Network
WSDL	Web Service Description Language
XHTML	Extensible Hypertext Markup Language
XML	eXtensible Markup Language
XML-RPC	XML Remote Procedure Call

Literaturverzeichnis

AWI05a Alfred-Wegener-Institut (Hrsg.): *Firmenprofil*. Version: 15. April 2005. http://www.awi-bremerhaven.de/AWI/index-d.html. – Online–Ressource, Abruf: 18. Juli 2005

AWI05b Alfred-Wegener-Institut (Hrsg.): *Organigramm*. Version: 30. Juni 2005. http://www.awi-bremerhaven.de/AWI/organigramm-d.html. – Online–Ressource, Abruf: 18. Juli 2005

Bal99a Balzert, Heide: *Lehrbuch der Objektmodellierung – Analyse und Entwurf*. München : Spektrum Akademischer Verlag, 1999. – ISBN 3–8274–1162–9

Bal99b Balzert, Helmut: *Lehrbuch Grundlagen der Informatik – Konzepte und Methoden in Java, C++ und UML, Algorithmik und Software-Technik*. München : Spektrum Akademischer Verlag, 1999. – ISBN 3–8274–0358–8

BLHL01 Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: *The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. Version: 17. Mai 2001. http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21. – Online–Ressource, Abruf: 27. Juli 2005

BMI02 Bundesministerium des Innern (Hrsg.): *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz*. Version: 24. Juli 2002. http://bundesrecht.juris.de/bundesrecht/bitv/. – Online–Ressource, Abruf: 15. August 2005

BOA04 Budapest Open Access Initiative (Hrsg.): *A Guide to Institutional Repository Software v 3.0*. Version: August 2004. http://www.soros.org/openaccess/software/. – Online–Ressource, Abruf: 18. Juli 2005

Cap97 Capelleveen, Dr. R. ; Universitätsbibliothek der Freien Universität Berlin (Hrsg.): *Zur Bedeutung von Metadaten für die Erschließung von Internetquellen*. Version: 06. Juni 1997. `http://www.ub.fu-berlin.de/~rvc/metadata/metadaten.html`. – Online–Ressource, Abruf: 31. Juli 2005

Car03 Carpenter, Leona ; The Open Archives Forum (Hrsg.): *OAI for Beginners - the Open Archives Forum online tutorial*. Version: 14. Oktober 2003. `http://www.oaforum.org/tutorial/`. – Online–Ressource, Abruf: 01. August 2005

CGMW03 Chinnici, Roberto ; Gudgin, Martin ; Moreau, Jean-Jacques ; Weerawarana, Sanjiva ; W3C (Hrsg.): *Web Services Description Language (WSDL) Version 1.2 - W3C Working Draft*. Version: 03. März 2003. `http://www.w3.org/TR/2003/WD-wsdl12-20030303/`. – Online–Ressource, Abruf: 17. August 2005

CJM05 Convissor, Daniel ; Jansen, Martin ; Merz, Alexander ; The PEAR Documentation Group (Hrsg.): *PEAR documentation*. Version: 18. September 2005. `http://pear.php.net/manual/en/`. – Online–Ressource, Abruf: 21. September 2005

DCM04 Dublin Core Metadata Initiative (Hrsg.): *Dublin Core Metadata Element Set, Version 1.1: Reference Description*. Version: 20. Dezember 2004. `http://dublincore.org/documents/dces/`. – Online–Ressource, Abruf: 25. Juli 2005

Don01 Donath, Andreas ; Klass & Ihlenfeld Verlag GmbH (Hrsg.): *Studie: Insellösungen bei eGovernment kosten Milliarden*. Version: 18. April 2001. `http://www.golem.de/0104/13500.html`. – Online–Ressource, Abruf: 08. September 2005

Don05 Donath, Andreas ; Klass & Ihlenfeld Verlag GmbH (Hrsg.): *PDF als Standard für Langzeitarchivierungen anerkannt*. Version: 18. September 2005. `http://www.golem.de/0509/40450.html`. – Online–Ressource, Abruf: 22. September 2005

Fed02 Fedora Development Team ; Cornell University (Hrsg.): *Mellon Fedora Technical Specification*. Version: 20. Dezember 2002. `http://www.fedora.info/documents/master-spec-12.20.02.pdf`. – Online–Ressource, Abruf: 06. September 2005

Fed05 Fedora Development Team ; Cornell University (Hrsg.): *Tutorial1: Introduction to Fedora*. Version: 28. Januar 2005. `http://www.fedora.info/download/2.0/userdocs/tutorials/tutorial1.pdf`. – Online–Ressource, Abruf: 18. Juli 2005

GHM+03 Gudgin, Martin ; Hadley, Marc ; Mendelsohn, Noah ; Moreau, Jean-Jacques ; Nielsen, Henrik F. ; W3C (Hrsg.): *SOAP Version 1.2 Part 1: Messaging Framework - W3C Recommendation.* Version: 24. Juni 2003. http://www.w3.org/TR/soap12-part1/. – Online–Ressource, Abruf: 17. August 2005

Jen04 Jendryschik, Michael: *Sprachen im Webauthoring.* Version: 4. September 2004. http://jendryschik.de/wsdev/einfuehrung/websprachen. – Online–Ressource, Abruf: 22. August 2005

KL04 Kossmann, Donald ; Leymann, Frank: Web Services. In: *Informatik-Spektrum* Band 27 (2004), April, Nr. 2, Seiten 117-128. http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s00287-004-0378-9

Kre04 Krebs, Kathleen: *Konzeption und Prototyp einer Client-/Server-Architektur für Online-Publishing basierend auf Webservicetechnologien.* http://vcbsup3.rrz.uni-hamburg.de/~rzdh008/diplom/data/diplom/diplom_finish.pdf. Version: 01. Juni 2004

Moz04 Mozilla Foundation (Hrsg.): *The Mozilla Public License, version 1.1.* Version: 08. November 2004. http://www.mozilla.org/MPL/. – Online–Ressource, Abruf: 25. Juli 2005

Mru02 Mruck, Katja: *BOAI FAQ auf deutsch beim Open Access Journal FQS (Forum Qualitative Sozialforschung/ Forum: Qualitative Social Research).* Version: 17. August 2002. http://www.qualitative-research.net/fqs/boaifaq.htm. – Online–Ressource, Abruf: 18. Juli 2005

Net05 Netcraft Ltd. (Hrsg.): *Web Server Survey August 2005.* Version: 1. August 2005. http://news.netcraft.com/archives/2005/08/01/web_server_survey_turns_10_finds_70_million_sites.html. – Online–Ressource, Abruf: 22. August 2005

OAI02 Open Archives Initiative (Hrsg.): *Open Archives Initiative Frequently Asked Questions (FAQ).* Version: 10. Juni 2002. http://www.openarchives.org/documents/FAQ.html. – Online–Ressource, Abruf: 27. Juli 2005

OAI04 Open Archives Initiative (Hrsg.): *The Open Archives Initiative Protocol for Metadata Harvesting 2.0.* Version: 12. Oktober 2004. http://www.openarchives.org/OAI/openarchivesprotocol.html. – Online–Ressource, Abruf: 18. Juli 2005

OMS05 Over, Albert ; Maiworm, Friedhelm ; Schelewski, André ; Deutsche Forschungsgemeinschaft, Bereich Informationsmanagement (Hrsg.): *Publikationsstrategien im Wandel? - Ergebnisse einer Umfrage zum Publikations- und Rezeptionsverhalten unter besonderer Berücksichtigung von Open Access*. Version: 21. Juli 2005. `http://www.dfg.de/dfg_im_profil/zahlen_und_fakten/statistisches_berichtswesen/open_access/download/oa_ber_dt.pdf`. – Online–Ressource, Abruf: 27. Juli 2005

PHP05 The PHP Group (Hrsg.): *PHP Manual*. Version: 14. Juli 2005. `http://www.php.net/manual/en/introduction.php`. – Online–Ressource, Abruf: 01. August 2005

Sie05 Sietmann, Richard ; Heise Zeitschriften Verlag (Hrsg.): *DFG legt Studie zu Open Access vor*. Version: 23. Juli 2005. `http://www.heise.de/newsticker/meldung/62004`. – Online–Ressource, Abruf: 27. Juli 2005

SUB01 Niedersächsische Staats- und Universitätsbibliothek Göttingen (Hrsg.): *Metaguide: Einführung in Metadaten*. Version: 27. März 2001. `http://www2.sub.uni-goettingen.de/intrometa.html`. – Online–Ressource, Abruf: 27. Juli 2005

Sub04 Suber, Peter: *A Very Brief Introduction to Open Access*. Version: 29. Dezember 2004. `http://www.earlham.edu/~peters/fos/brief.htm`. – Online–Ressource, Abruf: 27. Juli 2005

Tri05 Trinkwalder, Andrea ; Heise Zeitschriften Verlag (Hrsg.): *Langzeitarchivierung: PDF/A als ISO-Standard*. Version: 15. September 2005. `http://www.heise.de/newsticker/meldung/63957`. – Online–Ressource, Abruf: 22. September 2005

Wik05a Wikimedia Foundation Inc. (Hrsg.): *Begriff: Metadaten*. Version: 25. Juni 2005. `http://de.wikipedia.org/wiki/Metadaten`. – Online–Ressource, Abruf: 26. Juli 2005

Wik05b Wikimedia Foundation Inc. (Hrsg.): *Begriff: PHP*. Version: 12. August 2005. `http://de.wikipedia.org/wiki/Bild:PHP_funktionsweise.png`. – Online–Ressource, Abruf: 15. August 2005

Wik05c Wikimedia Foundation Inc. (Hrsg.): *Begriff: Repository*. Version: 14. Juli 2005. `http://en.wikipedia.org/wiki/Repository`. – Online–Ressource, Abruf: 18. Juli 2005

Wik05d Wikimedia Foundation Inc. (Hrsg.): *Begriff: Web Service*. Version: 01. März 2005. `http://de.wikipedia.org/wiki/Bild:Webservice.png`. – Online–Ressource, Abruf: 22. August 2005

Wik05e Wikimedia Foundation Inc. (Hrsg.): *Begriff: XHTML*. Version: 7. August 2005. `http://de.wikipedia.org/wiki/Xhtml`. – Online–Ressource, Abruf: 22. August 2005

Wik05f Wikimedia Foundation Inc. (Hrsg.): *Begriff: XML*. Version: 16. August 2005. `http://de.wikipedia.org/wiki/Xml`. – Online–Ressource, Abruf: 19. August 2005

WW98 Wolf, Misha ; Wicksteed, Charles ; W3C (Hrsg.): *Date and Time Formats (ISO 8601)*. Version: 27. August 1998. `http://www.w3.org/TR/NOTE-datetime`. – Online–Ressource, Abruf: 15. August 2005

Anhang

Anhang A

Quelltext: Konfiguration Fedora-Repository

A.1 fedora.fcfg

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://www.fedora.info/definitions/1/0/config/" xmlns:fedora-config="http://www.
    fedora.info/definitions/1/0/config/" class="fedora.server.BasicServer">
  <param name="repositoryName" value="Fedora at AWI"/>
  <comment>Defines a human readable name for the Fedora server; default is Fedora Repository.</
      comment>
  <param name="adminEmailList" value="fedora-admin@awi-bremerhaven.de"/>
  <comment>Defines one or more email addresses for server administrators; list is space delimited.</
      comment>
  <param name="adminPassword" value="**********"/>
  <comment>Defines the password for the Fedora server; default is fedoraAdmin.</comment>
  <param name="log_max_size" value="5242880"/>
  <comment>Defines the threshold at which a new log file is generated based on the size of the
      current log file in bytes; default is 5242880 bytes.</comment>
  <param name="log_max_days" value="7"/>
  <comment>Defines the threshold at which a new log file is generated based on number of days since
      the server was started in days; default is 7 days.</comment>
  <param name="log_dir" value="logs"/>
  <comment>Defines the directory in which the Fedora logs will be written; default is logs.</comment
      >
  <param name="log_level" value="info"/>
  <comment>Defines the level of logging desired;default is finest; possible levels include:
    - config
      denotes logging indicating what occurred during the server's (or a module's) configuration
          phase.
    - fine
      denotes logging indicating basic information about a request to the server (like hostname,
          operation name, and success or failure).
    - finer
      denotes logging indicating detailed information about a request to the server (like the full
          request, full response, and timing information).
    - finest (default)
      denotes logging indicating method entry/exit or extremely verbose information intended to aid
          in debugging.
    - info
```

```
26      denotes logging indicating that something relatively uncommon and interesting happened, like
            server or module startup or shutdown, or a periodic job.
27    - warning
28      denotes logging indicating that an undesired (but non-fatal) condition occured.
29    - severe
30      denotes logging indicating that the server is inoperable or unable to start.
31  </comment>
32  <param name="log_flush_threshold" value="1"/>
33  <comment>Defines the flush threshold for logging in number of records; default is one log record.<
        /comment>
34  <param name="path_algorithm" value="fedora.server.storage.lowlevel.TimestampPathAlgorithm"/>
35  <comment>Defines the java class used to determine the path algorithm for LowLevelStorage;
36    default is fedora.server.storage.lowlevel.TimestampPathAlgorithm.
37  </comment>
38  <param name="file_system" value="fedora.server.storage.lowlevel.GenericFileSystem"/>
39  <comment>Defines the java class that determines the type of implementation for LowLevelStorage;
40    default is fedora.server.storage.lowlevel.GenericFileSystem.
41  </comment>
42  <param name="registry" value="fedora.server.storage.lowlevel.DBPathRegistry"/>
43  <comment>Defines the java class used to determine the path registry for LowLevelStorage;
44    default is fedora.server.storage.lowlevel.DBPathRegistry.
45  </comment>
46  <param name="backslash_is_escape" value="yes" oraclevalue="no"/>
47  <comment>Defines whether the backing database (which include registry tables) the backslash
        character
48    to be the escape character (token beginning an escape sequence). This is needed to correctly
49    store and retrieve filepaths from the registry tables, if running under Windows/DOS.
50  </comment>
51  <param name="object_store_base" value="/ext/fedora/data/fedora2_0_objects"/>
52  <comment>Defines the root directory for the internal storage of Fedora objects.
53    This value should be adjusted based on your installation environment;
54    default assumes a Windows installation of C:\fedora2_0_objects.
55  </comment>
56  <param name="temp_store_base" value="/ext/fedora/data/fedora2_0_temp"/>
57  <comment>Defines the root directory for Fedora temporary storage.
58    This value should be adjusted based on your installation environment;
59    default assumes a Windows installation of C:\fedora2_0_temp.
60  </comment>
61  <param name="datastream_store_base" value="/ext/fedora/data/fedora2_0_datastreams"/>
62  <comment>Defines the root directory for the internal storage of Managed Content datastreams.
63    This value should be adjusted based on your installation environment;
64    default assumes a Windows installation of C:\fedora2_0_datastreams.
65  </comment>
66  <param name="debug" value="false"/>
67  <comment>A boolean toggle used to turn off/on internal debugging in the code.</comment>
68  <param name="datastreamExpirationLimit" value="300"/>
69  <comment>
70    Controls the size of the datastream mediation hash by removing entries outside the specified
          threshold.
71    The value is specified in seconds.
72    Note this value must be greater than the limit specified for the datastreamMediationLimit.
73  </comment>
74  <param name="datastreamMediationLimit" value="5000"/>
75  <comment>
76    Determines the time interval in which external mechanisms must respond to requests by the Fedora
          server.
77    The value is specified in milliseconds.
78    The value specified should be set high enough to allow for an average response time from any
          single external mechanisms.
79    Note this value must be less than the limit specified for the datastreamExpirationLimit.
80  </comment>
81  <param name="fedoraServerPort" value="8080"/>
82  <comment>Defines the port number on which the Fedora server runs; default is 8080.</comment>
83  <param name="fedoraShutdownPort" value="8005"/>
84  <comment>Defines the port number used to shutdown the Fedora sever; default is 8005.</comment>
85  <param name="fedoraRedirectPort" value="8443"/>
```

```
86    <comment>Defines the redirect port of the Fedora sever; default is 8443.</comment>
87    <param name="fedoraServerHost" value="web.awi-bremerhaven.de"/>
88    <comment>Defines the host name for the Fedora server, as seen from the outside world.</comment>
89    <module role="fedora.server.storage.DOManager" class="fedora.server.storage.DefaultDOManager">
90      <comment>
91          Description:
92            The interface to the storage subsystem. This provides
93            context-appropriate DOReaders and DOWriters for reflecting on and
94            writing to the objects stored in the repository. It also
95            provides methods for reflecting on the contents of the repository
96            as a whole. This implementation uses DefinitiveDOReader/Writer
97            for an "application" of "apim" and a FastDOReader/Writer
98            for an "application" of "apia" (from the Context). Other
99            context-specific logic TBD.
100         Parameters:
101         - pidNamespace (required)
102           This is the namespace id for pids of newly-created objects.
103           This should be unique for a repository.
104           It can be from 1 to 17 characters, and may only contain
105           A-Z, a-z, 0-9, and '-' (dash).
106         - retainPIDs [default="demo test"]
107           Namespaces of PIDs to retain during the ingest process.
108           When an object is ingested, Fedora normally allocates a unique PID
109           within pidNamespace for it regardless of what the object says its
110           PID is. This option provides a way to override that behavior on
111           a per-pid-namespace basis. If specified, this should be a
112           space-delimited list of pid namespaces that will be accepted in the
113           object as-is.
114         - storagePool [default=default provided by the ConnectionPoolManager]
115           The named connection pool from which read/write
116           database connections are to be provided for the storage
117           subsystem (see the ConnectionPoolManager module)
118         - storageCharacterEncoding [default=UTF-8]
119           If the serialization format is text-based,
120           this is the character encoding that should be used.</comment>
121         - defaultExportFormat (required)
122           The DEFAULT format in which external serializations of digital objects
123           are to be written. There must exist a correspondingly
124           named serializer-deserializer pair in the DOTranslator module.
125           Valid values are: "foxml1.0" and "metslikefedora1"
126     <param name="pidNamespace" value="awi"/>
127     <param name="retainPIDs" value="*"/>
128     <param name="storagePool" value="localMySQLPool" mckoivalue="localMcKoiPool" oraclevalue="
          localOracle9iPool"/>
129     <param name="storageCharacterEncoding" value="UTF-8"/>
130     <param name="defaultExportFormat" value="foxml1.0"/>
131   </module>
132   <module role="fedora.server.management.Management" class="fedora.server.management.
          DefaultManagement">
133     <comment>
134         Description:
135           The management subsystem.
136           This implements the methods necessary to fulfill API-M
137           requests without regard to:
138             - how the service is exposed
139             - how bytestreams and java types might be marshalled/demarshalled
140               over the wire
141             - how the storage subsystem is implemented.
142         Parameters:
143             - allowHosts (optional...to make unspecified, comment out or delete the whole param line
                    as opposed to using an empty string)
144               A comma-separated list of IP ranges that the client's address
145               is compared to. If this is specified, the remote address
146               must match for any Management request to be accepted.
147               If this is not specified, all requests will be accepted unless
148               the remote address matches a deny pattern.
```

```
149             - denyHosts (optional...to make unspecified, comment out or delete the whole param line
                    as opposed to using an empty string)
150               A comma-separated list of IP ranges that the client's address
151               is compared to. If this is specified, the remote address
152               must not match for any Management request to be accepted.
153               If this is not specified, request acceptance is governed
154               solely by the allowHosts parameter.
155         </comment>
156     <param name="allowHosts" value="127.0.0.1"/>
157   </module>
158   <module role="fedora.server.access.Access" class="fedora.server.access.DefaultAccess">
159     <comment>
160         Description:
161           The access subsystem.
162           This implements the methods necessary to fulfill API-A
163           requests without regard to:
164             - how the service is exposed
165             - how bytestreams and java types might be marshalled/demarshalled
166               over the wire
167             - how the storage subsystem is implemented.
168         Parameters:
169             - allowHosts (optional...to make unspecified, comment out or delete the whole param line
                    as opposed to using an empty string)
170               A comma-separated list of IP ranges that the client's address
171               is compared to. If this is specified, the remote address
172               must match for any Management request to be accepted.
173               If this is not specified, all requests will be accepted unless
174               the remote address matches a deny pattern.
175             - denyHosts (optional...to make unspecified, comment out or delete the whole param line
                    as opposed to using an empty string)
176               A comma-separated list of IP ranges that the client's address
177               is compared to. If this is specified, the remote address
178               must not match for any Management request to be accepted.
179               If this is not specified, request acceptance is governed
180               solely by the allowHosts parameter.
181             - doMediateDatastreams (required)
182               A boolean switch indicating whether Datastream Mediation is activated or not.
183               Datastream Mediation provides additional repository security by not exposing
184               the actual physical location of Referenced Content datastreams to external
185               mechanisms(services). Instead of exposing the actual physical location of
186               Referenced Content datastreams , Datastream Mediation functions as a proxy
187               requiring all external services to communicate through the Fedora server to
188               resolve the location of Referenced Content datastreams. For Datastream
189               Mediation to function, the Fedora server must be internet accessible on its
190               designated port by any potential external service. This requirement can be
191               difficult to satisfy if the Fedora server is located behind a firewall and
192               you do not have ready access to the firewall's configuration to enable the
193               required access. If the Fedora server is behind a firewall and you have no
194               access to the firewall's configuration, you can disable Datastream Mediation
195               which will allow external services to access Referenced Content datastreams
196               using their actual physical locations. Managed Content and XMLMetadata
197               datastreams are stored internally in the Fedora repository and can only be
198               accessed through the Fedora server so disabling Datastream Mediation has no
199               effect on these types of datastreams. Disabling Datastream Mediation exposes
200               the physical location of Referenced Content datastream to external services
201               which can result in unintended access to the raw datastreams by surreptitious
202               external mechanisms. It is recommended that Datastream Mediation be disabled
203               only for testing or cases where this security issue is not a concern. The
204               default value of doMediateDatastreams is true.
205             </comment>
206     <param name="doMediateDatastreams" value="true"/>
207   </module>
208   <module role="fedora.server.access.DynamicAccess" class="fedora.server.access.DynamicAccessModule"
        >
209     <comment>
210         Description:
```

```
211           The dynamic behavior module for the access subsystem.
212           This implements the methods necessary to fulfill API-A
213           requests without regard to:
214              - dynamically associating a default behavior definition and mechanism with objects
215             - (Future) dynamically associating other behavior definitions and mechanisms with objects
216             - running disseminations of dynamic behaviors
217         Parameters:
218             - fedora-system:1
219               (future) The interface that defines the methods of the bootstrap
220               disseminator. These methods are "built-in" to the Fedora
221               system, and are dynamically associated with every
222               behavior definition and behavior mechanism object.
223             - fedora-system:2
224               (future) The class that implements the methods of the bootstrap
225               disseminator. These method implementations are "built-in"
226               to the Fedora system, and are dynamically associated with
227               every behavior definition and behavior mechanism object.
228               This class can be though of as implementing
229               an "internal service" whereas other disseminators use
230               external services (described by WSDL) to do their work.
231             - fedora-system:3
232               The interface that defines the methods of the default
233               disseminator. These methods are "built-in" to the Fedora
234               system, and are dynamically associated with every object.
235             - fedora-system:4
236               The class that implements the methods of the default
237               disseminator. These method implementations are "built-in"
238               to the Fedora system, and are dynamically associated with
239               every object. This class can be though of as implementing
240               an "internal service" whereas other disseminators use
241               external services (described by WSDL) to do their work.
242           </comment>
243     <param name="fedora-system:1" value="fedora.server.access.internalservices.Bootstrap"/>
244     <param name="fedora-system:2" value="fedora.server.access.internalservices.BootstrapImpl"/>
245     <param name="fedora-system:3" value="fedora.server.access.defaultdisseminator.
            DefaultDisseminator"/>
246     <param name="fedora-system:4" value="fedora.server.access.defaultdisseminator.
            DefaultDisseminatorImpl"/>
247   </module>
248   <module role="fedora.server.search.FieldSearch" class="fedora.server.search.FieldSearchSQLModule">
249     <comment>
250         Description:
251           Supports the API-A simpleSearch and advancedSearch methods.
252         Parameters:
253           - maxResults (required, must be > 0)
254             The maximum number of records to return as the result of a search.
255             Even if a client requests more results at a time, this is the
256             cutoff value.
257           - maxSecondsPerSession (required, must be > 0)
258             The maximum number of seconds that the server guarantees subsequent
259             search results may be obtained. This is only used in cases where
260             the number of results is greater than maxResults (as specified
261             by the server [above] or the client [in the search request])
262           - connectionPool (optional...to make unspecified, comment out or delete the whole param
                line as opposed to using an empty string, default=ConnectionPoolManager's default)
263             The connectionPool providing the connection to the database to
264             be used.
265          Warning:
266             When setting these values, keep in mind that while a session
267             is not timed out (maxSecondsPerSession seconds haven't elapsed,
268             and not all results have been requested), a connection from the
269             pool is tied up. Therefore, the connectionPool should be at least
270             of size n, large enough to accomodate n simultaneous search
271             sessions. The longer maxSecondsPerSession is, the more chance
272             you have of tying up all available connections from the pool.
273             Therefore, keep maxSecondsPerSession fairly low, but still
```

```
274           reasonable for an automated program or user to serially get
275           a long list of results, and make sure you have a connectionPool
276           large enough to accomodate your users.
277         </comment>
278     <param name="maxResults" value="100"/>
279     <param name="maxSecondsPerSession" value="180"/>
280     <param name="connectionPool" value="localMySQLPool" mckoivalue="localMcKoiPool" oraclevalue="
            localOracle9iPool"/>
281   </module>
282   <module role="fedora.server.resourceIndex.ResourceIndex" class="fedora.server.resourceIndex.
          ResourceIndexModule">
283     <comment>
284         Description:
285           Supports the ResourceIndex.
286         Parameters:
287           - level (required)
288             Index level. Currently, only 0, 1 and 2 are supported levels.
289             0 = off: do not load the ResourceIndex
290             1 = basic: system metadata, RELS-EXT, disseminations
291             2 = basic + method permutations
292             WARNING: changing the level (except to 0) requires a full dump-and-load
293             of the repository.
294           - datastore (required)
295             Name of the triplestore to use.
296             WARNING: changing the triplestore requires a full dump-and-load
297             of the repository.
298         </comment>
299     <param name="level" value="0"/>
300     <param name="datastore" value="localKowariTriplestore"/>
301   </module>
302   <module role="fedora.oai.OAIProvider" class="fedora.server.oai.FedoraOAIProviderModule">
303     <comment>
304         Description:
305           Exposes the repository for OAI harvesters.</comment>
306         Parameters:
307           - repositoryName (required)
308             The name of the repository, to be given to OAI harvesters.
309           - repositoryDomainName (required)
310             The domain name of the repository, which helps to uniquely identify
311       items from the repository in OAI-PMH requests. This should just
312       be the domain name of the organization that exposes the objects.
313       More information on the OAI identifier syntax can be found at
314       http://www.openarchives.org/OAI/2.0/guidelines-oai-identifier.htm
315           - adminEmails (required)
316             One or more space-separated email addresses, to be given to OAI harvesters.
317           - friends (optional...to make unspecified, comment out or delete the whole param line as
                opposed to using an empty string)
318             A space-separated list of URLs to known OAI provider baseURLs.
319             This is OAI's provider discovery enabler. Harvesters can use these to crawl OAI providers
                  .
320           - maxRecords (optional...to make unspecified, comment out or delete the whole param line
                as opposed to using an empty string, default=FieldSearch:maxResults, imposed maximum=
                FieldSearch:maxResults)
321             The maximum number of results to return at a time for ListRecords requests.
322           - maxHeaders (optional...to make unspecified, comment out or delete the whole param line
                as opposed to using an empty string, default=FieldSearch:maxResults, imposed maximum=
                FieldSearch:maxResults)
323             The maximum number of results to return at a time for ListIdentifiers requests.
324     <param name="repositoryName" value="Fedora at AWI"/>
325     <param name="repositoryDomainName" value="awi-bremerhaven.de"/>
326     <param name="adminEmails" value="fedora-admin@awi-bremerhaven.de"/>
327     <param name="friends" value="http://ws.pangaea.de/oai/ http://zitmac05.gkss.de/fmi/xsl/oai/oai.
            xsl"/>
328     <param name="maxRecords" value="100"/>
329     <param name="maxHeaders" value="100"/>
330   </module>
```

```
331   <module role="fedora.server.storage.translation.DOTranslator" class="fedora.server.storage.
          translation.DOTranslatorModule">
332     <comment>Supports translation from DigitalObject to a stream of
333         some format, and vice-versa. The parameters below specify
334         serializer/deserializer classes to be used for a given format.
335         Those classes must implement the DOSerializer/DODeserializer
336         interfaces.
337         </comment>
338     <param name="serializer_metslikefedora1" value="fedora.server.storage.translation.
              METSLikeDOSerializer"/>
339     <param name="deserializer_metslikefedora1" value="fedora.server.storage.translation.
              METSLikeDODeserializer"/>
340     <param name="serializer_foxml1.0" value="fedora.server.storage.translation.FOXMLDOSerializer"/>
341     <param name="deserializer_foxml1.0" value="fedora.server.storage.translation.FOXMLDODeserializer
            "/>
342   </module>
343   <module role="fedora.server.management.PIDGenerator" class="fedora.server.management.
          BasicPIDGenerator">
344     <comment>The pid generator.</comment>
345     <param name="pidgen_log_dir" value="pidgen"/>
346   </module>
347   <module role="fedora.server.storage.replication.DOReplicator" class="fedora.server.storage.
          replication.DefaultDOReplicator">
348     <comment>Supports synching data from definitive storage with
349       dissemination db.</comment>
350   </module>
351   <module role="fedora.server.storage.ConnectionPoolManager" class="fedora.server.storage.
          ConnectionPoolManagerImpl">
352     <comment>This module facilitates obtaining ConnectionPools</comment>
353     <param name="defaultPoolName" value="localMySQLPool" mckoivalue="localMcKoiPool" oraclevalue="
            localOracle9iPool"/>
354     <param name="poolNames" value="localMySQLPool" mckoivalue="localMcKoiPool" oraclevalue="
            localOracle9iPool"/>
355   </module>
356   <module role="fedora.server.validation.DOValidator" class="fedora.server.validation.
          DOValidatorModule">
357     <comment>
358     Description:
359     Supports validation of digital objects, including XML Schema validation,
360     Schematron validation (to Fedora Rules schema written in Schematron language),
361     and other programatic validation including referential integrity checking of existence and
            availability of distributed data and/or services.
362     Parameters:
363       - xsd_metslikefedora1: this is local path for the Fedora-METS XML schema used to do XML schema
              validation of digital objects
364       - xsd_foxml1.0: this is local path for the Fedora FOXML XML schema used to do XML schema
            validation of digital objects
365       - rules_metslikefedora1: this is the local path to the Fedora Schematron Rules used to do
            Fedora-specific on Fedora-METS XML files.
366       - rules_foxml1.0: this is the local path to the Fedora Schematron Rules used to do Fedora-
            specific on FOXML XML files.
367       - schtron_preprocessor: this is the local path to the "skeleton" style sheet for schematron
            processing
368       - tempDir (required): this is a directory that the validation module can use as a work space,
            as when it must
369       write a temporary file.
370           </comment>
371     <param name="xsd_metslikefedora1" value="xsd/mets-fedora-ext.xsd"/>
372     <param name="xsd_foxml1.0" value="xsd/foxml1-0.xsd"/>
373     <param name="rules_metslikefedora1" value="schematron/metsExtRules1-0.xml"/>
374     <param name="rules_foxml1.0" value="schematron/foxmlRules1-0.xml"/>
375     <param name="schtron_preprocessor" value="schematron/preprocessor.xslt"/>
376     <param name="tempDir" value="work"/>
377   </module>
378   <module role="fedora.server.storage.ExternalContentManager" class="fedora.server.storage.
          DefaultExternalContentManager">
```

```
379     <comment>This module facilitates obtaining external content via HTTP
380               Parameters:
381               - userAgent (optional...to make unspecified, comment out or delete the whole param
                      line as opposed to using an empty string, default=Fedora): How to identify the
382                 Fedora server in HTTP requests. This defaults to something
383                 reasonable, but if you want to provide more information
384                 (such as an admin email address) in the User-Agent HTTP
385                 request header, put it here.
386               </comment>
387   </module>
388   <module role="fedora.server.utilities.ThreadMonitor" class="fedora.server.utilities.
          ThreadMonitorModule">
389     <comment>Prints basic information about the system memory and running threads to the
390            log periodically for diagnostic purposes.
391            Parameters:
392             - active (optional...to make unspecified, comment out or delete the whole param line as
                    opposed to using an empty string, default=no): whether to activate this module. "
                    yes" or "true" activates it.
393             - onlyMemory (optional, default=false) - only logs free memory, not thread details.
394             - pollInterval (optional...to make unspecified, comment out or delete the whole param
                    line as opposed to using an empty string, default=10000): how many milliseconds to
                    wait between each polling.
395            </comment>
396     <param name="active" value="false"/>
397     <param name="onlyMemory" value="false"/>
398     <param name="pollInterval" value="5000"/>
399   </module>
400   <datastore id="localMySQLPool">
401     <comment>MySQL database on localhost with db name of fedora</comment>
402     <param name="dbUsername" value="fedoraAdmin"/>
403     <param name="dbPassword" value="fedoraAdmin"/>
404     <param name="jdbcURL" value="jdbc:mysql://e-db.awi-bremerhaven.de/fedora?useUnicode=true&amp;
          characterEncoding=UTF-8&amp;autoReconnect=true"/>
405     <param name="jdbcDriverClass" value="com.mysql.jdbc.Driver"/>
406     <param name="ddlConverter" value="fedora.server.utilities.MySQLDDLConverter"/>
407     <param name="minPoolSize" value="10"/>
408     <param name="maxPoolSize" value="50"/>
409   </datastore>
410   <datastore id="localMcKoiPool">
411     <comment>McKoi database on localhost running on port 9157</comment>
412     <param name="dbUsername" value="fedora"/>
413     <param name="dbPassword" value="fedora"/>
414     <param name="jdbcURL" value="jdbc:mckoi://localhost:9157/"/>
415     <param name="jdbcDriverClass" value="com.mckoi.JDBCDriver"/>
416     <param name="ddlConverter" value="fedora.server.utilities.McKoiDDLConverter"/>
417     <param name="minPoolSize" value="10"/>
418     <param name="maxPoolSize" value="100"/>
419   </datastore>
420   <datastore id="localOracle9iPool">
421     <comment>Oracle9i database on localhost with SID=fedora20</comment>
422     <param name="dbUsername" value="fedoraAdmin"/>
423     <param name="dbPassword" value="fedoraAdmin"/>
424     <param name="jdbcURL" value="jdbc:oracle:thin:@localhost:1521:fedora20"/>
425     <param name="jdbcDriverClass" value="oracle.jdbc.driver.OracleDriver"/>
426     <param name="ddlConverter" value="fedora.server.utilities.OracleDDLConverter"/>
427     <param name="minPoolSize" value="10"/>
428     <param name="maxPoolSize" value="100"/>
429   </datastore>
430     <datastore id="localKowariTriplestore">
431         <comment>local Kowari Triplestore used by the Resource Index</comment>
432         <param name="connectorClassName" value="org.trippi.impl.kowari.KowariConnector"/>
433         <param name="remote" value="false"/>
434         <param name="path" value="/ext/fedora/data/fedora2_0_resourceIndex"/>
435         <param name="serverName" value="fedora"/>
436         <param name="modelName" value="ri"/>
437         <param name="poolInitialSize" value="3"/>
```

```
438         <param name="poolMaxGrowth" value="-1"/>
439         <param name="readOnly" value="false"/>
440         <param name="autoCreate" value="true"/>
441         <param name="autoTextIndex" value="true"/>
442         <param name="memoryBuffer" value="true"/>
443         <param name="autoFlushDormantSeconds" value="5"/>
444         <param name="autoFlushBufferSize" value="20000"/>
445         <param name="bufferFlushBatchSize" value="20000"/>
446         <param name="bufferSafeCapacity" value="40000"/>
447     </datastore>
448 </server>
```

Quelltext A.1: fedora.fcfg

Anhang B

Quelltext: Repository-Datenimport

B.1 LDAP2Fedora-Cronjob

B.1.1 ePIC2fedora.php

```php
<?php

/*************************/
/******** CONFIG *********/
/*************************/

$mode = 2;
// 1 = vor einfügen neuer datensätze wird überprüft, ob diese bereits vorhanden sind (zeitaufwändig ) und ggf. aktualisiert
// 2 = objekte werden ohne überprüfung auf änderungen in das fedora-repository importiert, bereits existente werden geskippt

// Ingestion begins with publications number greater than specified
$offset = 0;

// fedora server location
$fedoraServer = "web.awi-bremerhaven.de:8080";
$fedoraUsername = "fedoraAdmin";
$fedoraPassword = "fedoraAdmin";
$fedoraAPIA = "/fedora/access/soap?wsdl";
$fedoraAPIM = "/fedora/management/soap?wsdl";
$fedoraUploadInterface = "/fedora/management/upload";

// both include opening and closing foxml tags, DC can be directly inserted between them
$foxml_open = file_get_contents("ingesting/foxml_open.xml");
$foxml_close = file_get_contents("ingesting/foxml_close.xml");

// ldap server location (this is where the publications metadata is
$ldapHost = "ldap.awi-bremerhaven.de";
$ldapPort = "389";
$ldapUser = "";
$ldapPass = "";
$baseDN = "ou=Publications,dc=awi-bremerhaven,dc=de";

// these fields are read from the ldap-directory
$fields = array(
```

```
35      "createtimestamp",
36      "cruiseleg",
37      "discipline",
38      "modifytimestamp",
39      "platformdn",
40      "publicationauthor",
41      "publicationauthor;alternate",
42      "publicationawinumber",
43      "publicationdataset",
44      "publicationdoi",
45      "publicationemail",
46      "publicationidentifier",
47      "publicationsource",
48      "publicationtag",
49      "publicationtitle",
50      "publicationtype",
51      "publicationyear",
52      "sensor",
53      "thesisacceptedby",
54      "uid");
55
56  // path to the publications (searching for fulltext-versions)
57  $publication_fulltext_path = "http://www2.awi-bremerhaven.de/Publications/";
58  $fulltext = array();
59
60  /*************************/
61  /**** START OPERATION ****/
62  /*************************/
63
64  // connect to the ldap-directory
65  $ldap = ldap_connect($ldapHost,$ldapPort);
66  $conn = ldap_bind($ldap,$ldapUser,$ldapPass);
67
68  // split the results into years: otherwise the requested results may be too much to handle
69  // splitted into years <= 1979 and then goes in yearsteps up till present year
70  $years = array("<=1979");
71  for( $i = 1980; $i <= date("Y"); $i++ ) {
72    $years[] = "=".$i;
73  }
74
75  // for each year get all publications and ingest them into the fedora repository
76  $time_start = microtime("float");
77  $count_all_entries = 0;
78
79  foreach( $years as $year ) {
80    $time_year_start = microtime("float");
81
82    // set ldap-query-filter to get entrys for actual year
83    $filter = "(&(publicationYear".$year.")(publicationStatus=published))";
84
85    // execute ldap-query
86    $result = ldap_search($ldap,$baseDN,$filter,array_values($fields));
87    $entrys = ldap_get_entries($ldap, $result);
88    echo "publicationYear".$year.": <b>".$entrys["count"]."</b>\n";
89
90    // connect to fedora repository with acces-rights (API-A) and management-rights (API-M)
91    try { $soapclient_api_a = new SoapClient("http://".$fedoraServer.$fedoraAPIA,array("trace" => 1, "
          exceptions" => 1)); } catch(SoapFault $fault) { soap_error($soapclient_api_a,$fault); }
92    try { $soapclient_api_m = new SoapClient("http://".$fedoraUsername.":".$fedoraPassword."@".
          $fedoraServer.$fedoraAPIM,array("login" => $fedoraUsername, "password" => $fedoraPassword, "
          trace" => 1, "exceptions" => 1)); } catch(SoapFault $fault) { soap_error($soapclient_api_m,
          $fault); }
93
94    // begin transfer for each publication of the actual year
95    for( $i = 0; $i < $entrys["count"]; $i++) {
96      $count_all_entries++;
```

```
97    if( $count_all_entries > $offset ) {
98      $time_record_start = microtime("float");
99      echo " ".$count_all_entries." ".$year."(".$entrys["count"].") ".($i+1)." ".$uid." ";
100     if( $mode == 1 ) {
101       // in this mode records are checked of existance before they will be ingested
102       // existing objects are also checked on difference, in case of difference they will be
              updated
103       // $createtimestamp = $entrys[$i]["createtimestamp"][0];
104       // $modifytimestamp = $entrys[$i]["modifytimestamp"][0];
105       // $createtimestamp = strtotime(substr($createtimestamp,0,4)."-".substr($createtimestamp
              ,4,2)."-".substr($createtimestamp,6,2)."T".substr($createtimestamp,8,7));
106       // $modifytimestamp = strtotime(substr($modifytimestamp,0,4)."-".substr($modifytimestamp
              ,4,2)."-".substr($modifytimestamp,6,2)."T".substr($modifytimestamp,8,7));
107
108       // get identifier from epic-object
109       $uid = $entrys[$i]["uid"][0];
110
111       // baue verbindung zu fedora auf mit findObjects (api-a)
112       $arrayOfConditions = array(new SoapVar(new SoapVar(new condition("identifier","has",$uid),
              SOAP_ENC_OBJECT),SOAP_ENC_OBJECT, "Condition", "http://www.fedora.info/definitions/1/0/
              types/"));
113       $query = array(conditions => $arrayOfConditions, terms => null);
114       $response = $soapclient_api_a->findObjects(array("pid"),10,$query);
115
116       //überprüfe ob objekt existiert
117       if( $response->resultList ) {
118         // objekt existiert bereits
119         echo "<span style=\"color:#008800\">found</span>";
120
121         // nimm pid vom objekt aus fedora
122         $pid = $response->resultList[0]->pid;
123         echo ",pid:".$pid;
124
125         // hole daten mit getDatastream (api-m)
126         $response = $soapclient_api_m->getObjectXML($pid);
127         //var_dump(base64_decode($response));
128         $obj = simplexml_load_string(base64_decode($response));
129   //TODO:!!
130         var_dump($obj->children());
131         // $fedora_createtimestamp = strtotime($response->objCreateDate);
132         // $fedora_modifytimestamp = strtotime($response->objLastModDate);
133
134         // var_dump($fedora_createtimestamp);
135         // var_dump($fedora_modifytimestamp);
136
137         // vergleiche fed.objekt mit aktuellem objekt aus epic
138         if( $fedora_modifytimestamp < $modifytimestamp ) {
139           // wenn unterschied
140           $dcxml = prepare_entry($entrys[$i]);
141
142           // übersende dcxml mit modifyDatastreamByValue (api-m)
143           $response = $soapclient_api_m->modifyDatastreamByValue();
144         }
145       } else {
146         // objekt existiert noch nicht
147
148         echo "<span style=\"color:#FF0000\">not found</span>";
149
150         // build foxml and encode it with base64
151
152         //$foxml = str_replace("[--LASTMODIFIEDDATE--]",$modifytimestamp,str_replace("[--
              CREATEDDATE--]",$createtimestamp,$foxml_open))."\n";
153         $foxml = $foxml_open."\n";
154         $foxml .= $dctags;
155         $foxml .= $foxml_close;
156         $encodedfoxml = base64_encode($foxml);
```

```
157
158           // ingest object into fedora repository
159           $time_ingesting_start = microtime("float");
160           //$response = $soapclient_api_m->ingest($encodedfoxml,"foxml1.0","");
161           //echo $foxml;
162           $time_ingesting_end = microtime("float");
163           $time_ingesting = $time_ingesting_end - $time_ingesting_start;
164           echo number_format($time_ingesting*1000,0)."ms";
165         }
166       } else if( $mode == 2 ) { // in this mode records are note checked of existance, all found
            objects from the ldap-directory will be ingested, duplicate entrys will arise
167
168         // build foxml and encode it base64
169         echo "xml:";
170         $time_foxml_start = microtime("float");
171         $uid = str_replace(" ","%20",$entrys[$i]["uid"][0]);
172         $pid = "pub:".$uid;
173         $foxml = str_replace("[--PID--]"," PID=\"".$pid."\"",$foxml_open."\n");
174         $foxml .= prepare_entry($entrys[$i]);
175         $foxml .= $foxml_close;
176         $encodedfoxml = base64_encode($foxml);
177         $time_foxml_end = microtime("float");
178         $time_foxml = $time_foxml_end - $time_foxml_start;
179         echo number_format($time_foxml*1000,0)."ms\ting:";
180
181         // ingest object into fedora repository
182         $time_ingesting_start = microtime("float");
183         try {
184           $response = $soapclient_api_m->ingest($encodedfoxml,"foxml1.0","");
185         } catch(SoapFault $fault) {
186           //soap_error($soapclient_api_m,$fault);
187           echo "-FAIL-";
188         }
189         $time_ingesting_end = microtime("float");
190         $time_ingesting = $time_ingesting_end - $time_ingesting_start;
191         echo number_format($time_ingesting*1000,0)."ms";
192
193         // add fulltext-datastrem, if existing...
194         try {
195           if( !($obj = simplexml_load_string($foxml)) ) throw new exception();
196         } catch( exception $e ) {
197           echo "Parsing Error on uid:".$uid."\n".$e->getMessage();
198           var_dump($foxml);
199           continue;
200         }
201
202         $ns_foxml = $obj->children("info:fedora/fedora-system:def/foxml#");
203         $oai_dc = $ns_foxml->datastream->datastreamVersion->xmlContent->children("http://www.
              openarchives.org/OAI/2.0/oai_dc/");
204         $dc = $oai_dc->children("http://purl.org/dc/elements/1.1/");
205         foreach( $dc->format as $type ) {
206           $dsID = $fulltext["$type"]["id"];
207           $dsLabel = "Full text ".$dsID;
208           $dsLocation = $fulltext["$type"]["url"];
209           try {
210             $soapclient_api_m->addDatastream($pid,$dsID,array(""),$dsLabel,true,$type,null,
                    $dsLocation,"M","A","DS added via PHP");
211           } catch(SoapFault $fault) {
212             //soap_error($soapclient_api_m,$fault);
213             echo "--FAIL--\n";
214           }
215         }
216
217         // add datastream including some metadata from LDAP which are not already stored in the dc-
              datastream
218         // $more_metadata = "<metadata>\n"
```

```
219                 // ." <createtimestamp>".$entrys[$i]["createtimestamp"][0]."</createtimestamp>\n"
220                 // ." <modifytimestamp>".$entrys[$i]["modifytimestamp"][0]."</modifytimestamp>\n"
221                 // ."</metadata>\n";
222
223         // POST /fedora/management/upload HTTP/1.1
224         // Authorization: Basic ZmVkb3JhQWRtaW46ZmVkb3JhQWRtaW4=
225         // User-Agent: Jakarta Commons-HttpClient/2.0.1
226         // Host: web.awi-bremerhaven.de:1234
227         // Content-Length: 390
228         // Content-Type: multipart/form-data; boundary=----------------31415926535897932384 6
229
230         // ------------------31415926535897932384 6
231         // Content-Disposition: form-data; name="file"; filename="fedora-upload-62681.tmp"
232         // Content-Type: application/octet-stream; charset=ISO-8859-1
233         // Content-Transfer-Encoding: binary
234
235         // <metadata>
236          // <createtimestamp>20041214174250Z</createtimestamp>
237          // <modifytimestamp>20041214174250Z</modifytimestamp>
238         // </metadata>
239         // ------------------31415926535897932384 6--
240
241         // $post_data = array("cc"=>"us \n");
242         // $post_data[variable_1] = $more_metadata;
243         // $post_data[filename] = "@tmp.txt";
244         // $curl = curl_init("http://".$fedoraUsername.":".$fedoraPassword."@".$fedoraServer.
                $fedoraUploadInterface);
245         // curl_setopt($curl, CURLOPT_VERBOSE, 1);
246         // curl_setopt($curl, CURLOPT_POST, 1);
247         // curl_setopt($curl, CURLOPT_POSTFIELDS, $post_data);
248         // curl_exec($curl);
249         // curl_close($curl);
250
251         // $dsID = "MORE_META";
252         // $dsLabel = "Additional Metadata";
253         // $dsLocation = "";
254         // try {
255           // $soapclient_api_m->addDatastream($pid,$dsID,array(""),$dsLabel,true,$type,null,
                $dsLocation,"X","A","DS added via PHP");
256         // } catch(SoapFault $fault) {
257           // soap_error($soapclient_api_m,$fault);
258           // echo "--FAIL--\n";
259         // }
260       }
261       $time_record_end = microtime("float");
262       $time_record = $time_record_end - $time_record_start;
263       echo "\tall:".number_format($time_record,3)."sec\n";
264     }
265   }
266   $time_year_end = microtime("float");
267   $time_year = $time_year_end - $time_year_start;
268   echo "Time for ".$year.": ".number_format($time_year,2)."s (".number_format($entrys["count"]/
        $time_year,1)."records/sec) (".number_format($time_year/$entrys["count"],3)."sec/record)\n";
269 }
270 $time_end = microtime("float");
271 $time = $time_end - $time_start;
272 echo "Time for all: ".number_format($time,2)."s (".number_format($count_all_entries/$time,1)."
      records/s) (".number_format($time/$count_all_entries,3)."s/record)\n";
273
274 ldap_close($ldap);
275
276 // FÜR JEDES OBJEKT TUE: {
277   // hole identifier von epic-objekt
278   // baue verbindung zu fedora auf mit findObjects (api-a)
279   // überprüfe ob objekt existiert
280   // wenn nicht existiert: {
```

```
281     // wandle in foxml
282     // ingest (api-m)
283   // }
284   // wenn existiert: {
285     // nimm pid vom objekt aus fedora
286     // hole daten mit getDatastream (api-m)
287     // vergleiche fed.objekt mit aktuellem objekt aus epic
288     // wenn unterschied: {
289       // formatiere dcxml
290       // übersende dcxml mit modifyDatastreamByValue (api-m)
291     // }
292   // }
293 // }
294
295 function soap_error ($soapclient,$fault) {
296   // prints human-readable SOAP-Error
297   echo "\nfaultcode:( $fault->faultcode )\nfaultstring:( $fault->faultstring )\n";
298   // echo "<h2>findObjects Request</h2>\n";
299   // $xml = DOMDocument::loadXML($soapclient->__getLastRequest());
300   // $xml->formatOutput = true;
301   // echo "<pre>".htmlspecialchars($xml->saveXML(), ENT_QUOTES)."</pre>\n"
302     // ."<h2>findObjects Response</h2>\n";
303   // $xml = DOMDocument::loadXML($soapclient->__getLastResponse());
304   // $xml->formatOutput = true;
305   // echo "<pre>".htmlspecialchars($xml->saveXML(), ENT_QUOTES)."</pre>\n";
306 }
307
308 class condition {
309    function condition($p, $o, $v)
310    {
311        $this->property = $p;
312        $this->operator = new SoapVar($o,null,"ComparisonOperator","http://www.fedora.info/
               definitions/1/0/types/");
313        $this->value = $v;
314    }
315 }
316
317 function prepare_entry($entry) {
318   // builds dcxml from ldap-entry
319
320   global $publication_fulltext_path,$fulltext;
321
322   //----------------------------
323   // retrieve data from entrys
324   //----------------------------
325   $createtimestamp = $entry["createtimestamp"][0];
326   $cruiseleg = $entry["cruiseleg"];
327   $disciplines = $entry["discipline"];
328   $modifytimestamp = $entry["modifytimestamp"][0];
329   $platformdns = $entry["platformdn"];
330   $publicationauthor = $entry["publicationauthor"][0];
331   $publicationcoauthors = $entry["publicationauthor;alternate"];
332   $publicationawinumber = $entry["publicationawinumber"][0];
333   $publicationdataset = $entry["publicationdataset"][0];
334   $publicationdoi = $entry["publicationdoi"][0];
335   $publicationemail = $entry["publicationemail"][0];
336   $publicationidentifier = $entry["publicationidentifier"][0];
337   $publicationsource = $entry["publicationsource"][0];
338   $publicationtag = $entry["publicationtag"];
339   $publicationtitle = $entry["publicationtitle"][0];
340   $publicationtypes = $entry["publicationtype"];
341   $publicationyear = $entry["publicationyear"][0];
342   $sensors = $entry["sensor"];
343   $thesisacceptedby = $entry["thesisacceptedby"][0];
344   $uid = $entry["uid"][0];
345
```

```
346   $todays_date = date("Y-m-d",time());
347
348   //----------------------------
349   // prepare data for XML-file
350   //----------------------------
351
352   // datestamp, extractes from ldap-timestamp
353   $datestamp=substr($createtimestamp,0,4)."-".substr($createtimestamp,4,2)."-".substr(
          $createtimestamp,6,2);
354
355   // modifydatestamp
356   $modifydatestamp=substr($modifytimestamp,0,4)."-".substr($modifytimestamp,4,2)."-".substr(
          $modifytimestamp,6,2);
357
358   // if doi exists, create tagged string
359   $doi_string="";
360   if ($publicationdoi != "") {
361     $doi_string="<dc:identifier>doi:$publicationdoi</dc:identifier>";
362     $doi_string_description="<dc:description>doi:$publicationdoi</dc:description>";
363   }
364
365   // if publicationawinumber exists, create tagged string
366   $publicationawinumber_string="";
367   if (($publicationawinumber != "") && ($publicationawinumber != "none") ) {
368     $publicationawinumber_string="<dc:identifier>$publicationawinumber</dc:identifier>";
369   }
370
371   // make array from authors and coauthors and put it into a string
372   $creator_string="";
373   $creator_string=" <dc:creator>".htmlspecialchars($publicationauthor,ENT_NOQUOTES,"UTF-8")."</dc:
          creator>\n";
374   for( $j = 0; $j < $publicationcoauthors["count"]; $j++ ) {
375     $creator_string.=" <dc:creator>".htmlspecialchars($publicationcoauthors[$j],ENT_NOQUOTES,"UTF-8"
           )."</dc:creator>\n";
376   }
377
378   // compilation for publisher
379   $publisher_string = "";
380   if ($publicationsource != "") {
381     $publisher_string.=$publicationsource.", ";
382   }
383   if ($thesisacceptedby != "") {
384     $publisher_string.=$thesisacceptedby.", ";
385   }
386   if ($publicationyear != "") {
387     $publisher_string.=$publicationyear;
388   }
389
390   // @publicationtypes
391   $type_string = "";
392   for( $j = 0; $j < $publicationtypes["count"]; $j++ ) {
393     $type_string.=" <dc:type>$publicationtypes[$j]</dc:type>\n";
394   }
395
396   // fulltext
397   $format_string="";
398   $fulltext = array(
399     "text/html" => array("id" => "HTML", "url" => $publication_fulltext_path.$uid.".html"),
400     "application/pdf" => array("id" => "PDF", "url" => $publication_fulltext_path.$uid.".pdf"),
401     "application/postscript" => array("id" => "PS", "url" => $publication_fulltext_path.$uid.".ps"),
402     "application/ppt" => array("id" => "PPT", "url" => $publication_fulltext_path.$uid.".ppt")
403   );
404   $ch = curl_init();
405   curl_setopt($ch, CURLOPT_HEADER, 1);
406   curl_setopt($ch, CURLOPT_NOBODY, 1);
407   curl_setopt($ch, CURLOPT_VERBOSE, 0);
```

```
408   curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
409   curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
410   foreach( $fulltext as $type => $source ) {
411     curl_setopt($ch, CURLOPT_URL, $source["url"]);
412     $http_response = curl_exec($ch);
413     if( strpos($http_response,"HTTP/1.1 200 OK") !== false ) {
414       $format_string .= " <dc:format>".$type."</dc:format>\n";
415     }
416     // if( @fopen($source["url"],"r") ) {
417       // $format_string .= " <dc:format>".$type."</dc:format>\n";
418     // }
419   }
420   curl_close($ch);
421
422   // publicationdataset
423   $relation_string="";
424   if ($publicationdataset != "") {
425     $relation_string="<dc:relation>uri:$publicationdataset</dc:relation>\n";
426   }
427
428   // prepare dc:subject which is compilated by publicationtag ( AWIgroupname,lang-en),
429   //discipline and a
430   //commalist: PlatformID (namedisplay from Ou=Platforms), cruiseleg, sensor
431   $subject_string="";
432   for( $j = 0; $j < $publicationtag["count"]; $j++ ) {
433     $orgunitname=getvalue("cn=".$publicationtag[$j],"ou=Groups,dc=awi-bremerhaven,dc=de","
          AWIgroupname;lang-en");
434     if( $orgunitname ) {
435       $subject_string.=" <dc:subject>".htmlspecialchars($orgunitname[0],ENT_NOQUOTES,"UTF-8")."</dc:
            subject>\n";
436     }
437   }
438   for( $j = 0; $j < $disciplines["count"]; $j++ ) {
439     $subject_string.=" <dc:subject>".htmlspecialchars($disciplines[$j],ENT_NOQUOTES,"UTF-8")."</dc:
          subject>\n";
440   }
441   $platform_substring="";
442   for( $j = 0; $j < $platformdns["count"]; $j++ ) {
443     $arr = explode(",",$platformdns[0]);
444     $platformname=getvalue($arr[0],$platformdns[$j],"namedisplay");
445     $platform_substring.=$platformname[0].", ";
446   }
447   for( $j = 0; $j < $cruiseleg["count"]; $j++ ) {
448     $platform_substring.=$cruiseleg[$j].", ";
449   }
450   for( $j = 0; $j < $sensors["count"]; $j++ ) {
451     $platform_substring.=$sensors[$j].", ";
452   }
453   if ( $platform_substring != "" ) {
454     $platform_substring = rtrim($platform_substring,", "); // removes any ", " at the end of the
          string
455     $subject_string.=" <dc:subject>".htmlspecialchars($platform_substring,ENT_NOQUOTES,"UTF-8")."</
          dc:subject>\n";
456   }
457
458   #----------------------------
459   # build record metadata
460   #----------------------------
461   $a_record_as_string="";
462
463   $a_record_as_string.=" <oai_dc:dc\n";
464   $a_record_as_string.=" xmlns:oai_dc=\"http://www.openarchives.org/OAI/2.0/oai_dc/\"\n";
465   $a_record_as_string.=" xmlns:dc=\"http://purl.org/dc/elements/1.1/\""; //\n";
466   $a_record_as_string.=">\n";
467   if ($doi_string != "") {
468     $a_record_as_string.=" $doi_string\n";
```

```
469   }
470   $a_record_as_string.=" <dc:identifier>$uid</dc:identifier>\n";
471   $a_record_as_string.=" <dc:identifier>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/
          abstract.php?uid=$uid</dc:identifier>\n";
472   if ($publicationawinumber_string != "") {
473     $a_record_as_string.=" $publicationawinumber_string\n";
474   }
475   $a_record_as_string.=" <dc:title>".htmlspecialchars($publicationtitle,ENT_NOQUOTES,"UTF-8")."</dc:
          title>\n";
476   if ($creator_string != "") {
477     $a_record_as_string.="$creator_string";
478   }
479   $a_record_as_string.=" <dc:publisher>".htmlspecialchars($publisher_string,ENT_NOQUOTES,"UTF-8")."
          </dc:publisher>\n";
480   if ($type_string != "") {
481     $a_record_as_string.="$type_string";
482   }
483   if ($format_string != "") {
484     $a_record_as_string.="$format_string";
485   }
486   if ($relation_string != "") {
487     $a_record_as_string.=" $relation_string";
488   }
489   if ($subject_string != "") {
490     $a_record_as_string.="$subject_string";
491   }
492   if ($doi_string_description != "") {
493     $a_record_as_string.=" $doi_string_description\n";
494   }
495   $a_record_as_string.=" <dc:description>uri:http://www.awi-bremerhaven.de/php/PublicationAbstracts/
          abstract.php?uid=$uid</dc:description>\n";
496   $a_record_as_string.=" <dc:date>$publicationyear-01-01</dc:date>\n";
497   $a_record_as_string.=" <dc:date.modified>$modifydatestamp</dc:date.modified>\n";
498   $a_record_as_string.=" <dc:rights>uri:http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode</
          dc:rights>\n";
499   $a_record_as_string.=" </oai_dc:dc>\n";
500
501   return utf8_encode(utf8_decode($a_record_as_string));
502 }
503
504 function getvalue($dn,$base,$field) {
505   global $ldap;
506   $result = ldap_search($ldap,$base,$dn,array($field));
507   $entrys = ldap_get_entries($ldap, $result);
508   if( $entrys["count"] == 0 ) {
509     return;
510   } else {
511     return $entrys[0][strtolower($field)];
512   }
513 }
514
515 ?>
```

Quelltext B.1: ePIC2fedora.php

B.1.2 foxml_open.xml

```
1 <foxml:digitalObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xmlns:foxml="info:fedora/fedora-system:def/foxml#"
3   xsi:schemaLocation="info:fedora/fedora-system:def/foxml# http://www.fedora.info/definitions/1/0/
        foxml1-0.xsd"[--PID--]>
```

```
 <foxml:objectProperties>
  <foxml:property NAME="http://www.w3.org/1999/02/22-rdf-syntax-ns#type" VALUE="FedoraObject"/>
 </foxml:objectProperties>
 <foxml:datastream CONTROL_GROUP="X" ID="DC" STATE="A" VERSIONABLE="true">
  <foxml:datastreamVersion ID="DC1.0" LABEL="Dublin Core Metadata" MIMETYPE="text/xml">
   <foxml:xmlContent>
```

Quelltext B.2: foxml_open.xml

B.1.3 foxml_close.xml

```
    </foxml:xmlContent>
   </foxml:datastreamVersion>
  </foxml:datastream>
</foxml:digitalObject>
```

Quelltext B.3: foxml_close.xml

Anhang C

Quelltext: SOAP-Client

C.1 Interfaces

C.1.1 interface.webService.php

```php
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * interface.webService.php
 * @package Diplomarbeit
 */

/**
 * Web Service Interface
 *
 * Provides information about web service implementation
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
     works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */
interface webService {

  /**
   * Contructor
   *
   * @param String $name The name of the Web Service
```

```
36      * @param String $wsdl URL to web services' wsdl-file
37      */
38     public function __construct($name,$wsdl);
39
40     /**
41      * getSessionID
42      *
43      * Gets the actual ID identifying last searchrequest
44      *
45      * @return String The ID of the current session
46      */
47     public function getSessionID();
48
49     /**
50      * simpleSearch
51      *
52      * Performs a simplyfied search on the repository the web service is implemented in.
53      *
54      * @param String $terms Words to perform a searchrequest
55      * @param int $numberOfResults Listlength of the searchresponse
56      * @return String Formatted XML containig searchresults or errormessage
57      */
58     public function simpleSearch($terms,$numberOfResults);
59
60     /**
61      * moreResults
62      *
63      * If there are more results than the specified listlength set in simple search or advanced search
            , this function returns further results
64      *
65      * @param String $session Identifies the used session
66      * @param int $numberOfResults Listlength of the searchresponse
67      * @param offset $numberOfResults Listlength of the searchresponse
68      * @return String Formatted XML containig searchresults or errormessage
69      */
70     public function moreResults($session,$numberOfResults,$offset);
71
72     /**
73      * detailsForObject
74      *
75      * Gets detailed information from one object
76      *
77      * @param String $pid Identifies the requested object
78      * @return String Formatted XML containig searchresults or errormessage
79      */
80     public function detailsForObject($pid);
81 }
82
83 ?>
```

Quelltext C.1: interface.webService.php

C.2 Objektklassen

C.2.1 class.fedoraWebService.php

```
1  <?php
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
```

```
4
5  /**
6   * class.fedoraWebService.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * class fedoraWebService
12  *
13  * Implementation for Fedora-like Web Services. Tested with Fedora-Repository version 2.0.
14  * Connects the Repository via PHP:SOAP (requires PHP 5).
15  *
16  * PHP version 5
17  *
18  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
19  * Under the following conditions:
20  * You must attribute the work in the manner specified by the author or licensor.
21  * For any reuse or distribution, you must make clear to others the license terms of this work.
22  * Any of these conditions can be waived if you get permission from the copyright holder.
23  * Your fair use and other rights are in no way affected by the above.
24  *
25  * @package Helmholtz_WS
26  * @author Bastian Onken <bonken@awi-bremerhaven.de>
27  * @copyright 2005, Bastian Onken
28  * @license http://creativecommons.org/licenses/by/2.0/
29  * @version 1.0
30  */
31 final class fedoraWebService implements webService {
32
33   /**
34    * Contructor
35    *
36    * @param String $name The name of the Web Service
37    * @param String $wsdl URL to web services' wsdl-file
38    */
39   function __construct($name,$wsdl) {
40     $this->name = $name;
41     $this->wsdl = $wsdl;
42     $this->session = "";
43     $this->fields_thumbnail = array(
44       "title",
45       "creator",
46       "date",
47       "type",
48       "pid"
49     );
50     $this->fields_detailed = array(
51       "title",
52       "creator",
53       "subject",
54       "description",
55       "publisher",
56       "contributor",
57       "date",
58       "type",
59       "format",
60       "identifier",
61       "source",
62       "language",
63       "relation",
64       "coverage",
65       "rights",
66       "pid"
67     );
68   }
```

```
69
70    /**
71     * getSessionID
72     *
73     * Gets the actual ID identifying last searchrequest
74     *
75     * @return String The ID of the current session
76     */
77    function getSessionID() {
78      return $this->session;
79    }
80
81    /**
82     * simpleSearch
83     *
84     * Performs a simplyfied search on the repository where the web service is implemented.
85     *
86     * @param String $terms Words to perform a searchrequest
87     * @param int $numberOfResults Listlength of the searchresponse
88     * @return String Formatted XML containig searchresults or errormessage
89     */
90    function simpleSearch($terms,$numberOfResults) {
91      try {
92        /* Performing a 'findObjects'-SOAP-request: retrieve WSDL, build query, send SOAP-envelope,
              return response */
93        $soapclient = new SoapClient($this->wsdl,array("trace" => 1,"exceptions" => 1));
94        $query = array("conditions" => null,"terms" => "*".utf8_encode($terms)."*");
95        $response = $soapclient->findObjects($this->fields_thumbnail,$numberOfResults,$query);
96        $this->session = isset($response->listSession->token) ? $response->listSession->token : "";
97        return $this->map($response->resultList);
98      } catch(SoapFault $fault) {
99        /* SOAP Error-Handling: builds an XML-file; root-element is named 'search', soap-error is
              attached as child */
100       $dom = new DOMDocument('1.0', 'iso-8859-1');
101       $root = $dom->createElement("search");
102       $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
103       $root->appendChild($error);
104       $dom->appendChild($root);
105       var_dump($dom->saveXML());
106       return $dom->saveXML();
107     }
108   }
109
110   /**
111    * moreResults
112    *
113    * If there are more results than the specified listlength which is set through simple search,
           this function returns further results
114    *
115    * @param String $session Identifies the used session
116    * @param int $numberOfResults Listlength of the searchresponse
117    * @param offset $numberOfResults Listlength of the searchresponse
118    * @return String Formatted XML containig searchresults or errormessage
119    */
120   function moreResults($session,$numberOfResults,$offset) {
121     try {
122       /* Performing a 'resumeFindObjects'-SOAP-request: retrieve WSDL, send SOAP-envelope, return
              response */
123       $soapclient = new SoapClient($this->wsdl,array("trace" => 1, "exceptions" => 1));
124       $response = $soapclient->resumeFindObjects($session);
125       $this->session = isset($response->listSession->token) ? $response->listSession->token : "";
126       return $this->map($response->resultList);
127       /* SOAP Error-Handling: builds an XML-file; root-element is named 'search', soap-error is
              attached as child */
128     } catch(SoapFault $fault) {
129       $dom = new DOMDocument('1.0', 'iso-8859-1');
```

```
130       $root = $dom->createElement("search");
131       $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
132       $root->appendChild($error);
133       $dom->appendChild($root);
134       return $dom->saveXML();
135     }
136   }
137
138   /**
139    * detailsForObject
140    *
141    * Gets detailed information from one object
142    *
143    * @param String $pid Identifies the requested object
144    * @return String Formatted XML containig searchresults or errormessage
145    */
146   function detailsForObject($pid) {
147     if( !empty($pid) && !is_null($pid) ) {
148       try {
149         /* Performing a 'findObjects'-SOAP-request: retrieve WSDL, build query for this pid, send
                SOAP-envelope, return response */
150         $soapclient = new SoapClient($this->wsdl,array("trace" => 1, "exceptions" => 1));
151         $arrayOfConditions = array();
152         array_push($arrayOfConditions,new SoapVar(new SoapVar(new condition("pid","eq",utf8_encode(
                $pid)),SOAP_ENC_OBJECT),SOAP_ENC_OBJECT, "Condition", "http://www.fedora.info/
                definitions/1/0/types/"));
153         $query = array("conditions" => $arrayOfConditions, "terms" => null);
154         $response = $soapclient->findObjects($this->fields_detailed,1,$query);
155         return $this->map($response->resultList);
156       } catch(SoapFault $fault) {
157         /* SOAP Error-Handling: builds an XML-file; root-element is named 'search', soap-error is
                attached as child */
158         $dom = new DOMDocument('1.0', 'iso-8859-1');
159         $root = $dom->createElement("search");
160         $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
161         $root->appendChild($error);
162         $dom->appendChild($root);
163         return $dom->saveXML();
164       }
165     }
166   }
167
168   /**
169    * map
170    *
171    * Maps SOAP-Responses to a standardized XML document (the architecture of the XML file is equal
           to all further Web Service implementations)
172    *
173    * @param String $soap_response SOAP-Response of the performed Request
174    * @return String Formatted XML containig searchresults
175    */
176   private function map($soap_response) {
177     $dom = new DOMDocument("1.0", "iso-8859-1");
178     $root = $dom->createElement("search");
179     foreach( $soap_response as $res ) {
180       $element = $dom->createElement("result");
181       $item = array();
182       $item["dc_title"] = isset($res->title) ? $res->title : null;
183       $item["repository"] = "Fedora at AWI";
184       $item["dc_creator"] = isset($res->creator) ? $res->creator : null;
185       $item["dc_date"] = isset($res->date) ? substr($res->date,0,4) : null;
186       $item["dc_description"] = isset($res->description) ? $res->description : null;
187       $item["dc_subject"] = isset($res->subject) ? $res->subject : null;
188       $item["dc_coverage"] = isset($res->coverage) ? $res->coverage : null;
189       $item["dc_publisher"] = isset($res->publisher) ? $res->publisher : null;
190       $item["dc_contributor"] = isset($res->contributor) ? $res->contributor : null;
```

```
191       $item["dc_source"] = isset($res->source) ? $res->source : null;
192       $item["dc_language"] = isset($res->language) ? $res->language : null;
193       $item["dc_relation"] = isset($res->relation) ? $res->relation : null;
194       $item["dc_type"] = isset($res->type) ? $res->type : null;
195       $item["dc_format"] = isset($res->format) ? $res->format : null;
196       $item["dc_rights"] = isset($res->rights) ? $res->rights : null;
197       $item["dc_identifier"] = isset($res->pid) ? $res->pid : null;
198       foreach( $item as $key => $data ) {
199         if( is_array($data ) ) {
200           foreach( $data as $val ) {
201             $element->appendChild($dom->createElement($key,$val));
202           }
203         } else {
204           if( !is_null($item[$key]) ) {
205             $element->appendChild($dom->createElement($key,$data));
206           }
207         }
208       }
209       $root->appendChild($element);
210     }
211     $dom->appendChild($root);
212     return $dom->saveXML();
213   }
214 }
215
216 ?>
```

Quelltext C.2: class.fedoraWebService.php

C.2.2 class.pangaeaWebService.php

```
1  <?php
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5  /**
6   * class.pangaeaWebService.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * class pangaeaWebservice
12  *
13  * Implementation for Pangaea-like Web Services.
14  * Connects the Repository via PHP:SOAP (requires PHP 5).
15  *
16  * PHP version 5
17  *
18  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
19  * Under the following conditions:
20  * You must attribute the work in the manner specified by the author or licensor.
21  * For any reuse or distribution, you must make clear to others the license terms of this work.
22  * Any of these conditions can be waived if you get permission from the copyright holder.
23  * Your fair use and other rights are in no way affected by the above.
24  *
25  * @package Helmholtz_WS
26  * @author Bastian Onken <bonken@awi-bremerhaven.de>
27  * @copyright 2005, Bastian Onken
28  * @license http://creativecommons.org/licenses/by/2.0/
29  * @version 1.0
```

```
30  */
31 final class pangaeaWebService implements webService {
32
33   /**
34    * Contructor
35    *
36    * @param String $name The name of the Web Service
37    * @param String $wsdl URL to web services' wsdl-file
38    */
39   function __construct($name,$wsdl) {
40     $this->name = $name;
41     $this->wsdl = $wsdl;
42     $this->session = "";
43     $this->offset = 0;
44   }
45
46   /**
47    * getSessionID
48    *
49    * Gets the actual ID identifying last searchrequest
50    *
51    * @return String The ID of the current session
52    */
53   function getSessionID() {
54     return $this->session;
55   }
56
57   /**
58    * simpleSearch
59    *
60    * Performs a simplyfied search on the repository the web service is implemented in.
61    *
62    * @param String $terms Words to perform a searchrequest
63    * @param int $numberOfResults Listlength of the searchresponse
64    * @return String Formatted XML containig searchresults or errormessage
65    */
66    function simpleSearch($terms,$numberOfResults) {
67     try {
68       $soapclient = new SoapClient($this->wsdl,array("trace" => 1, "exceptions" => 1));
69       $query = $terms;
70       $minLat = -90;
71       $maxLat = 90;
72       $minLon = -180;
73       $maxLon = 180;
74       // $minLat = -180;
75       // $minLon = -90;
76       // $maxLat = 180;
77       // $maxLon = 90;
78       $offset = 0;
79       $this->session = $soapclient->registerSession(null, "", "", "PangaVista");
80       $response = $soapclient->search($this->session, iconv("ISO-8859-1","UTF-8",$query), $minLat,
             $minLon, $maxLat, $maxLon, $offset, $numberOfResults);
81       if( $response->totalCount < $numberOfResults ) $this->session = "";
82       return $this->map($response->results);
83     } catch(SoapFault $fault) {
84       $dom = new DOMDocument('1.0', 'iso-8859-1');
85       $root = $dom->createElement("search");
86       $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
87       $root->appendChild($error);
88       $dom->appendChild($root);
89       return $dom->saveXML();
90     }
91   }
92
93   /**
94    * moreResults
```

```
95      *
96      * If there are more results than the specified listlength set in simple search or advanced search
            , this function returns further results
97      *
98      * @param String $session Identifies the used session
99      * @param int $numberOfResults Listlength of the searchresponse
100     * @param offset $numberOfResults Listlength of the searchresponse
101     * @return String Formatted XML containig searchresults or errormessage
102     */
103    function moreResults($session,$numberOfResults,$offset) {
104      try {
105        $soapclient = new SoapClient($this->wsdl,array("trace" => 1, "exceptions" => 1));
106        $query = $terms;
107        $minLat = -180;
108        $minLon = -90;
109        $maxLat = 180;
110        $maxLon = 90;
111        $response = $soapclient->search($session, iconv("ISO-8859-1","UTF-8",$query), $minLat, $minLon
               , $maxLat, $maxLon, $offset, $numberOfResults);
112        $this->session = $session;
113        $this->offset = $offset;
114        if( $response->totalCount < ($numberOfResults+$offset) ) $this->session = "";
115        return $this->map($response->results);
116      } catch(SoapFault $fault) {
117        $dom = new DOMDocument('1.0', 'iso-8859-1');
118        $root = $dom->createElement("search");
119        $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
120        $root->appendChild($error);
121        $dom->appendChild($root);
122        return $dom->saveXML();
123      }
124    }
125
126    /**
127     * detailsForObject
128     *
129     * Gets detailed information from one object
130     *
131     * @param String $pid Identifies the requested object
132     * @return String Formatted XML containig searchresults or errormessage
133     */
134    function detailsForObject($pid) {
135      if( !empty($pid) && !is_null($pid) ) {
136        try {
137          $soapclient = new SoapClient($this->wsdl,array("trace" => 1, "exceptions" => 1));
138          $this->session = $soapclient->registerSession(null, $_SERVER['HTTP_USER_AGENT'], $_SERVER['
                 REMOTE_HOST'], "PangaVista");
139          $response[] = $soapclient->metadata($this->session,$pid);
140          return $this->map($response);
141        } catch(SoapFault $fault) {
142          $dom = new DOMDocument('1.0', 'iso-8859-1');
143          $root = $dom->createElement("search");
144          $error = $dom->createElement("soaperror",$this->name." Error: ".$fault->faultcode);
145          $root->appendChild($error);
146          $dom->appendChild($root);
147          return $dom->saveXML();
148        }
149      }
150    }
151
152    /**
153     * map
154     *
155     * Maps SOAP-Responses to a standardized XML document (the architecture of the XML file is equal
            to all further Web Service implementations)
156     *
```

```
157     * @param String $soap_response SOAP-Response of the performed request
158     * @return String Formatted XML containig searchresults
159     */
160    private function map($soap_response) {
161      $dom = new DOMDocument('1.0', 'iso-8859-1');
162      $root = $dom->createElement("search");
163      foreach($soap_response as $xml_item) {
164        // echo "<strong>RESULT:</strong><br/>";
165        // debug(htmlentities(base64_decode($xml_item)));
166        if( is_object($xml_item) ) {
167          $xml = simplexml_load_string(base64_decode($xml_item->xml));
168        } else {
169          $xml = simplexml_load_string(base64_decode($xml_item));
170        }
171        $namespace = 'http://www.pangaea.de/MetaData';
172        $item = array();
173        $item["dc_title"] = null;
174        $item["repository"] = "Pangaea";
175        $item["dc_creator"] = null;
176        $item["dc_date"] = null;
177        $item["dc_description"] = null;
178        $item["dc_subject"] = null;
179        $item["dc_coverage"] = null;
180        $item["dc_publisher"] = null;
181        $item["dc_contributor"] = null;
182        $item["dc_source"] = null;
183        $item["dc_language"] = null;
184        $item["dc_relation"] = null;
185        $item["dc_type"] = null;
186        $item["dc_format"] = null;
187        $item["dc_rights"] = null;
188        $item["dc_identifier"] = null;
189        $element = $dom->createElement("result");
190        foreach( $xml->children($namespace) as $key => $res ) {
191          if( $key == "citation" ) {
192            // echo "<strong> citation:</strong><br/>";
193            // debug($res->author);
194            $item["dc_title"] = (array)$res->title;
195            $item["dc_date"] = (array)$res->date;
196            $item["dc_source"] = (array)("uri:http://doi.pangaea.de/".(string)$res->URI);
197            $item["dc_publisher"] = (array)"uri:http://www.pangaea.de";
198            $item["dc_language"] = (array)"en";
199            $item["dc_type"] = (array)"Dataset";
200            $item["dc_rights"] = (array)"Copyright";
201            $item["dc_identifier"] = (array)$res->URI;
202            $item["dc_creator"] = array();
203            $arr_author = $res->author;
204            foreach( $arr_author as $author ) {
205              $item["dc_creator"][] = (string)$author->lastName.", ".(string)$author->firstName;
206            }
207            // $item["dc_description"] = $res->description;
208            // $item["dc_subject"] = $res->subject;
209          }
210          if( $key == "size" ) {
211            $item["dc_format"] = (string)$res;
212          }
213          if( $key == "extent" ) {
214            $west = round((float)$res->geographic->westBoundLongitude,4);
215            $east = round((float)$res->geographic->eastBoundLongitude,4);
216            $south = round((float)$res->geographic->southBoundLatitude,4);
217            $north = round((float)$res->geographic->northBoundLatitude,4);
218            $item["dc_coverage"] = (array)("WEST: ".$west." * EAST: ".$east." * SOUTH: ".$south." *
                   NORTH: ".$north);
219          }
220          if( $key == "reference" ) {
221            // if( !isset($item["dc_contributor"]) ) $item["dc_contributor"] = array();
```

```
222           if( !isset($item["dc_relation"]) ) $item["dc_relation"] = array();
223           $arr_author = $res->author;
224           foreach( $arr_author as $author ) {
225             // $item["dc_contributor"][] = (string)$author->lastName.", ".$author->firstName;
226             $string_authors .= (string)$author->lastName.", ".$author->firstName.", ";
227           }
228           $item["dc_relation"][] = rtrim($string_authors,", ")." (".$res->date."): ".$res->title.",
                  ".str_replace("& ","&amp; ",$res->source).", ".$res->pages;
229         }
230         if( $key == "technicalInfo" ) {
231           // TODO!!! better use xpath here!!!
232           foreach( $res->entry as $entry ) {
233             $get_next_value = false;
234             $parent = "";
235             foreach( $entry->attributes() as $tag => $attribute ) {
236               $arr_tag = (array)$tag;
237               $arr_attribute = (array)$attribute;
238               // echo $arr_tag[0]."<strong>".$arr_attribute[0]."</strong> ";
239               if( $get_next_value ) {
240                 if( $parent = "mimeType" ) $item["dc_format"] = $arr_attribute[0].", ".$item["
                        dc_format"];
241               }
242               if ($arr_attribute[0] == "mimeType") {
243                 $get_next_value = true;
244               }
245               $parent = $arr_attribute[0];
246             }
247           }
248         }
249       }
250       foreach( $item as $key => $data ) {
251         if( !is_null($data) ) {
252           if( is_array($data) ) {
253             foreach( $data as $val ) {
254               $element->appendChild($dom->createElement($key,$val));
255             }
256           } else {
257             $element->appendChild($dom->createElement($key,$data));
258           }
259         }
260       }
261       $root->appendChild($element);
262     }
263     $dom->appendChild($root);
264     return $dom->saveXML();
265   }
266 }
267
268 ?>
```

Quelltext C.3: class.pangaeaWebService.php

C.2.3 class.condition.php

```
1 <?php
2
3 /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5 /**
6  * class.condition.php
7  * @package Diplomarbeit
```

```
 */

/**
 * class condition
 *
 * This class is used to create the complex SOAP variable 'condition' (used in class.
     fedoraWebService.php)
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
     works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */
class condition {

  /**
   * Contructor
   *
   * @param String $p Property: this is the field which is searched on the given value (e.g. "mdate
       ")
   * @param String $o Operator (e.g. "has")
   * @param String $v Value (e.g. "atlantic")
   */
  function __construct($p, $o, $v) {
    $this->property = $p;
    $this->operator = new SoapVar($o,null,"ComparisonOperator","http://www.fedora.info/definitions
        /1/0/types/");
    $this->value = $v;
  }
}

?>
```

Quelltext C.4: class.condition.php

C.3 Wrapper

C.3.1 webServiceCaller.php

```
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * webServiceCaller.php
 * @package Diplomarbeit
 */

```

```php
/**
 * Web Service Caller
 *
 * Calls one Web Service by executing a remote search-operation
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */

require_once("includes/functions-common.inc.php");
require_once("includes/config.inc.php");
require_once("includes/xhtmlbuilders.inc.php");

// SIMPLE/ADVANCED SEARCH
if( isset($_GET["classid"]) && isset($_GET["mode"]) && ($_GET["mode"] == "simpleSearch" || $_GET["mode"] == "advancedSearch") ) {
  $class = $useable_webServices[$_GET["classid"]]["class"];
  $name = $useable_webServices[$_GET["classid"]]["name"];
  $wsdl = $useable_webServices[$_GET["classid"]]["wsdl"];
  $ws = new $class($name,$wsdl);
  $terms = (isset($_GET["terms"]) && !is_null($_GET["terms"])) ? $_GET["terms"] : "";
  $time_response_start = microtime("float");
  if( $_GET["mode"] == "simpleSearch" )
    $xml = $ws->simpleSearch($terms,$defaultNumberOfResultsDisplayed);
  if( $_GET["mode"] == "advancedSearch" )
    $xml = $ws->advancedSearch($_GET["conditions"],$defaultNumberOfResultsDisplayed);
  $time_response_end = microtime("float");
  $time_response = $time_response_end - $time_response_start;
  if( strpos($xml,"</search>") !== false ) {
    //String was found
    $output = str_replace("</search>","<technicalinfo><ws>".$useable_webServices[$_GET["classid"]]["name"]."</ws><wsid>".$_GET["classid"]."</wsid><session>".$ws->getSessionID()."</session><offset>0</offset><responsetime>".number_format($time_response,3)."s</responsetime></technicalinfo></search>",$xml);
  } else {
    $output = str_replace("<search/>","<search><technicalinfo><ws>".$useable_webServices[$_GET["classid"]]["name"]."</ws><wsid>".$_GET["classid"]."</wsid><session>".$ws->getSessionID()."</session><offset>0</offset><responsetime>".number_format($time_response,3)."s</responsetime></technicalinfo></search>",$xml);
  }
  echo $output;
}

// MORE RESULTS
if( isset($_GET["classid"]) && isset($_GET["sessionID"]) && isset($_GET["mode"]) && $_GET["mode"] == "moreResults" ) {
  $class = $useable_webServices[$_GET["classid"]]["class"];
  $name = $useable_webServices[$_GET["classid"]]["name"];
  $wsdl = $useable_webServices[$_GET["classid"]]["wsdl"];
  $ws = new $class($name,$wsdl);
  $terms = (isset($_GET["terms"]) && !is_null($_GET["terms"])) ? $_GET["terms"] : "";
  $time_response_start = microtime("float");
  $xml = $ws->moreResults($_GET["sessionID"],$defaultNumberOfResultsDisplayed,$_GET["offset"]);
```

```
67    $time_response_end = microtime("float");
68    $time_response = $time_response_end - $time_response_start;
69    if( !empty($xml) ) {
70      if( strpos($xml,"</search>") !== false ) {
71        //String was found
72        $output = str_replace("</search>","<technicalinfo><ws>".$useable_webServices[$_GET["classid"
              ]]["name"]."</ws><wsid>".$_GET["classid"]."</wsid><session>".$ws->getSessionID()."</
              session><offset>".$_GET["offset"]."</offset><responsetime>".number_format($time_response
              ,3)."s</responsetime></technicalinfo></search>",$xml);
73      } else {
74        $output = str_replace("<search/>","<search><technicalinfo><ws>".$useable_webServices[$_GET["
              classid"]]["name"]."</ws><wsid>".$_GET["classid"]."</wsid><session>".$ws->getSessionID().
              "</session><offset>".$_GET["offset"]."</offset><responsetime>".number_format(
              $time_response,3)."s</responsetime></technicalinfo></search>",$xml);
75      }
76    } else {}
77    echo $output;
78  }
79
80  ?>
```

Quelltext C.5: webServiceCaller.php

C.4 Funktionsbausteine

C.4.1 functions-search.inc.php

```
1   <?php
2
3   /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5   /**
6    * functions-search.inc.php
7    * @package Diplomarbeit
8    */
9
10  /**
11   * Functions to search repositories
12   *
13   * Contains one function to perfom a multithreaded search
14   *
15   * PHP version 5
16   *
17   * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
         works; to make commercial use of the work
18   * Under the following conditions:
19   * You must attribute the work in the manner specified by the author or licensor.
20   * For any reuse or distribution, you must make clear to others the license terms of this work.
21   * Any of these conditions can be waived if you get permission from the copyright holder.
22   * Your fair use and other rights are in no way affected by the above.
23   *
24   * @package Helmholtz_WS
25   * @author Bastian Onken <bonken@awi-bremerhaven.de>
26   * @copyright 2005, Bastian Onken
27   * @license http://creativecommons.org/licenses/by/2.0/
28   * @version 1.0
29   */
30
31  /**
```

```
32  * multiSearch
33  *
34  * Performs a simplyfied search on the repository where the web service is implemented.
35  *
36  * @param String $mode Words to perform a searchrequest
37  * @param String $terms Words to perform a searchrequest
38  * @param String $advancedConditions Words to perform a searchrequest
39  * @param String $sessions Words to perform a searchrequest
40  * @param Array $classids Listlength of the searchresponse
41  * @param String $offset Listlength of the searchresponse
42  * @return String Formatted XML containig searchresults or errormessage
43  */
44 function multiSearch($mode,$terms,$advancedConditions,$sessions,$classids,$offset) {
45   global $defaultNumberOfResultsDisplayed,$useable_webServices;
46   $connomains = array();
47   $i = 0;
48   $self = $_SERVER["PHP_SELF"];
49   $webapp_path = "http://".$_SERVER["HTTP_HOST"].":".$_SERVER["SERVER_PORT"].rtrim($self,strrchr(
         $self,"/"));
50   foreach( $classids as $classid ) {
51     $url = $webapp_path."/webServiceCaller.php"
52       ."?mode=".$mode
53       ."&classid=".$classid
54       ."&terms=".urlencode($terms);
55     $url .= !is_null($advancedConditions) ? "&conditions=".$advancedConditions : "";
56     $url .= isset($sessions[$i]) ? "&sessionID=".$sessions[$i] : "";
57     $url .= "&offset=".$offset;
58     $connomains[] = $url;
59     $i++;
60   }
61
62   $mh = curl_multi_init();
63
64   foreach ($connomains as $i => $url) {
65     $conn[$i] = curl_init($url);
66     curl_setopt($conn[$i], CURLOPT_RETURNTRANSFER, 1);
67     curl_setopt($conn[$i], CURLOPT_CONNECTTIMEOUT, $connection_timeout);
68     curl_setopt($conn[$i], CURLOPT_TIMEOUT, $response_timeout);
69     curl_multi_add_handle ($mh,$conn[$i]);
70   }
71
72   /* start performing the request */
73   do {
74     $mrc = curl_multi_exec($mh, $active);
75   } while ($mrc == CURLM_CALL_MULTI_PERFORM);
76
77   while ($active and $mrc == CURLM_OK) {
78     /*wait for network*/
79     if (curl_multi_select($mh) != -1) {
80      /*pull in any new data, or at least handle timeouts*/
81      do {
82     $mrc = curl_multi_exec($mh, $active);
83      } while ($mrc == CURLM_CALL_MULTI_PERFORM);
84     }
85   }
86
87   if ($mrc != CURLM_OK) {
88     print "Curl multi read error $mrc\n";
89   }
90
91   /*retrieve data*/
92   foreach ($connomains as $i => $url) {
93     if (($err = curl_error($conn[$i])) == '') {
94      $res[$i]=curl_multi_getcontent($conn[$i]);
95     } else {
96      $dom = new DOMDocument('1.0', 'iso-8859-1');
```

```
97      $root = $dom->createElement("error",$webServices[$i]["name"]." error: $err");
98      $dom->appendChild($root);
99      $res[$i]=$dom->saveXML();
100    }
101    curl_multi_remove_handle($mh,$conn[$i]);
102    curl_close($conn[$i]);
103   }
104   curl_multi_close($mh);
105   return $res;
106 }
107
108 ?>
```

Quelltext C.6: functions-search.inc.php

C.4.2 functions-common.inc.php

```
1  <?php
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5  /**
6   * functions-common.inc.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * Common functions
12  *
13  * Including general functions
14  *
15  * PHP version 5
16  *
17  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
18  * Under the following conditions:
19  * You must attribute the work in the manner specified by the author or licensor.
20  * For any reuse or distribution, you must make clear to others the license terms of this work.
21  * Any of these conditions can be waived if you get permission from the copyright holder.
22  * Your fair use and other rights are in no way affected by the above.
23  *
24  * @package Helmholtz_WS
25  * @author Bastian Onken <bonken@awi-bremerhaven.de>
26  * @copyright 2005, Bastian Onken
27  * @license http://creativecommons.org/licenses/by/2.0/
28  * @version 1.0
29  */
30
31 /**
32  * __autoload
33  *
34  * Automatically imports classes/interfaces located in 'includes/classes/' when they are requested
35  *
36  * @param String $name Name of Class/Interface
37  * @return boolean 'true' if import was successful, otherwise 'false'
38  */
39 function __autoload($name) {
40   $class = $interface = false;
41   // interface first
42   if (include_exists("includes/classes/interface.$name.php")) {
43     require_once("includes/classes/interface.$name.php");
```

```
44      $interface = true;
45    }
46    // class later
47    if (include_exists("includes/classes/class.$name.php")) {
48      require_once("includes/classes/class.$name.php");
49      $class = true;
50    }
51    // catch error
52    if( $class === false && $interface === false ) {
53      trigger_error("Failed to include class/interface $name",E_USER_WARNING);
54      return false;
55    }
56    return true;
57  }
58
59  /**
60   * include_exists
61   *
62   * Checks if an include exists (referenced from __autoload)
63   *
64   * @param String $file Path to Filename where Class/Interface is located
65   * @return boolean 'true' if class/interface was found, otherwise 'false'
66   */
67  function include_exists($file) {
68    static $include_dirs = null;
69    static $include_path = null;
70    // set include_dirs
71    if( is_null($include_dirs) || get_include_path() !== $include_path ) {
72      $include_path = get_include_path();
73      foreach( split(PATH_SEPARATOR,$include_path) as $include_dir ) {
74        if( substr($include_dir,-1) != '/' ) {
75          $include_dir .= '/';
76        }
77        $include_dirs[] = $include_dir;
78      }
79    }
80    if (substr($file,0,1) == '/') { //absolute filepath - what about file:///?
81      return (file_exists($file));
82    }
83    if( (substr($file, 0, 7) == 'http://' || substr($file, 0, 6) == 'ftp://') && ini_get('
          allow_url_fopen') ) {
84      return true;
85    }
86    foreach( $include_dirs as $include_dir ) {
87      if( file_exists($include_dir.$file) ) {
88        return true;
89      }
90    }
91    return false;
92  }
93
94  ?>
```

Quelltext C.7: functions-common.inc.php

C.5 Einstellungen

C.5.1 config.inc.php

```php
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * config.php
 * @package Diplomarbeit
 */

/**
 * Configuration for Helmholtz_WS
 *
 * Global settings
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */

/* PHP settings */
/* Set non-verbose error-reposrting */
error_reporting(E_ALL ^ E_NOTICE);
/* Suppress 'Cannot clone object of class DOMDocument' while using SimpleXML */
ini_set("zend.ze1_compatibility_mode","0");

/* Helmholtz_WS settings */
/* Specifies the number of searchresults from each WS */
$defaultNumberOfResultsDisplayed = "5";
/* Multithreading: Timeout when connecting service (seconds) */
$connection_timeout = 2;
/* Multithreading: service has to fullfill the request within this time (seconds) */
$response_timeout = 8;
/* Web Services to which an request is sent to (class must be implemented) */
$useable_webServices = array(
   array("class" => "fedoraWebService", "name" => "Fedora at AWI", "wsdl" => "http://web.awi-bremerhaven.de:8080/fedora/access/soap?wsdl")
  ,array("class" => "pangaeaWebService", "name" => "Pangaea", "wsdl" => "http://ws.pangaea.de/ws/services/PangaVista?wsdl")
  //,array("class" => "fedoraWebService", "name" => "tufts.edu", "wsdl" => "http://dl.tufts.edu:8080/fedora/access/soap?wsdl")
  //,array("class" => "fedoraWebService", "name" => "Arrowprod", "wsdl" => "http://arrowprod.lib.monash.edu.au:8080/fedora/access/soap?wsdl")
);

?>
```

Quelltext C.8: config.inc.php

C.6 GUI

C.6.1 header.inc.php

```php
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * header.inc.php
 * @package Diplomarbeit
 */

/**
 * Page header / global function import
 *
 * Generates XHTML-Header and includes all PHP-references (functions, classes, config, builders). Included in every output.
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */

require_once("functions-common.inc.php");
require_once("functions-search.inc.php");
require_once("config.inc.php");
require_once("xhtmlbuilders.inc.php");

readfile("includes/xhtmlheader.inc.html");

echo " <body>\n";

if( strpos($_SERVER["PHP_SELF"],"index.php") == false ) {
  echo " <div style=\"width:702px;margin:10px auto 5px auto\"><a href=\"index.php\"><img src=\" includes/images/banner.jpg\" style=\"border:1px solid #000000\" alt=\"Helmholtz Open Access\" /></a></div>\n";
}

?>
```

Quelltext C.9: header.inc.php

C.6.2 footer.inc.php

```php
<?php
```

```
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5  /**
6   * footer.inc.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * Page footer
12  *
13  * Generates XHTML (navigation, W3C validation images). Included in every output.
14  *
15  * PHP version 5
16  *
17  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
18  * Under the following conditions:
19  * You must attribute the work in the manner specified by the author or licensor.
20  * For any reuse or distribution, you must make clear to others the license terms of this work.
21  * Any of these conditions can be waived if you get permission from the copyright holder.
22  * Your fair use and other rights are in no way affected by the above.
23  *
24  * @package Helmholtz_WS
25  * @author Bastian Onken <bonken@awi-bremerhaven.de>
26  * @copyright 2005, Bastian Onken
27  * @license http://creativecommons.org/licenses/by/2.0/
28  * @version 1.0
29  */
30
31 echo " <div id=\"footer\">\n"
32     ." <div style=\"padding-bottom:15px\">| <a accesskey=\"s\" href=\"index.php\">start</a> | <a
          accesskey=\"l\" href=\"links.php\">links</a> | <a accesskey=\"a\" href=\"about.php\">about
          </a> | <a accesskey=\"h\" href=\"help.php\">help</a><!--<img src=\"includes/images/help.gif
          \" alt=\"Need Help?\" style=\"vertical-align:center;border:0\" />--> |</div>\n"
33     ." <a accesskey=\"X\" href=\"http://validator.w3.org/check?uri=referer\" title=\"W3 Consortium
          Document Validator\"><img src=\"includes/images/valid-xhtml11_lightgray.png\" alt=\"Valid
          XHTML 1.1!\" style=\"border:0\" width=\"80px\" height=\"15px\" /></a> |\n"
34     ." <a accesskey=\"C\" href=\"http://jigsaw.w3.org/css-validator/check/referer/\" title=\"W3
          Consortium Style Sheet Validator\"><img src=\"includes/images/valid-css_lightgray.png\" alt
          =\"Valid CSS!\" style=\"border:0\" width=\"80px\" height=\"15px\" /></a> |\n"
35     ." <a accesskey=\"W\" href=\"http://www.w3.org/WAI/WCAG1AAA-Conformance\" title=\"Explanation of
           Level Triple-A Conformance\"><img src=\"includes/images/wcag3A_lightgray.png\" alt=\"
          Triple-A Conformance!\" style=\"border:0\" width=\"80px\" height=\"15px\" /></a>\n"
36     ." </div>\n"
37     ." </body>\n"
38     ."</html>";
39
40 ?>
```

Quelltext C.10: footer.inc.php

C.6.3 xhtmlheader.inc.html

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
2    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
4   <head profile="http://dublincore.org/documents/dcq-html/">
5    <title>Repository Search</title>
6    <link rel="stylesheet" href="includes/styles/global.css" type="text/css" />
7    <link rel="schema.DC" href="http://purl.org/dc/elements/1.1/" />
```

```
8   <link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />
9   <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
10  <meta http-equiv="content.language" content="en" />
11  <meta name="robots" content="index,follow" />
12  <meta name="author" content="Bastian Onken" />
13  <meta name="description" content="AWI Web Services for primary data, publications and personal
        portfolio" />
14  <meta name="keywords" content="AWI, Alfred, Wegener, Institute, Fedora, SOAP, client, data,
        dataset, datasets, publication, publications, personal, portfolio" />
15  <meta name="DC.creator" content="Bastian Onken" />
16  <meta name="DC.title" content="Repository SOAP Client" />
17  <meta name="DC.subject" content="AWI Respository, webapplication for searching digital objects" />
18  <meta name="DC.description" content="AWI Web Services for primary data, publications and personal
        portfolio" />
19  <meta name="DC.publisher" content="Alfred Wegener Institute for polar and marine research" />
20  <meta name="DC.contributor" content="Ana Macario" />
21  <meta name="DC.date" scheme="DCTERMS.W3CDTF" content="2005-09-26T12:00:00+02:00" />
22  <meta name="DC.type" scheme="DCTERMS.DCMIType" content="Collection" />
23  <meta name="DC.format" scheme="DCTERMS.IMT" content="text/html" />
24  <meta name="DC.identifier" scheme="DCTERMS.URI" content="http://web.awi-bremerhaven.de/php/
        Helmholtz_WS/" />
25  <meta name="DC.source" content="http://web.awi-bremerhaven.de/php/Helmholtz_WS/index.php" />
26  <meta name="DC.language" scheme="DCTERMS.RFC3066" content="en" />
27  <meta name="DC.relation" content="diploma thesis" />
28  <meta name="DC.coverage" scheme="DCTERMS.TGN" content="Bremerhaven, Germany" />
29  <meta name="DC.rights" content="http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode" />
30 </head>
```

Quelltext C.11: xhtmlheader.inc.html

C.6.4 xhtmlbuilders.inc.php

```
1  <?php
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5  /**
6   * xhtmlbuilders.inc.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * Different XHTML generators
12  *
13  * These functions are used to display the searchresults and other XHTML
14  *
15  * PHP version 5
16  *
17  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
18  * Under the following conditions:
19  * You must attribute the work in the manner specified by the author or licensor.
20  * For any reuse or distribution, you must make clear to others the license terms of this work.
21  * Any of these conditions can be waived if you get permission from the copyright holder.
22  * Your fair use and other rights are in no way affected by the above.
23  *
24  * @package Helmholtz_WS
25  * @author Bastian Onken <bonken@awi-bremerhaven.de>
26  * @copyright 2005, Bastian Onken
27  * @license http://creativecommons.org/licenses/by/2.0/
28  * @version 1.0
```

```
 */

/**
 * debug
 *
 * Prints contents of a variable with HTML-safe output.
 *
 * @param mixed $val Variable to debug
 */
function debug($val) {
  echo "<pre>\n";
  var_dump($val);
  echo "</pre>\n";
}

/**
 * print_var
 *
 * Parses and returns the contents of a variable (array / SimpleXML-object). HTML-safe output.
 *
 * @param mixed $var Variable to print
 * @return String XHTML output
 */
function print_var($var=null) {
  if( is_array($var) ) {
    $arr = $var;
    if( $arr ) {
      $arr = array_unique($arr);
      //sort($arr);
      $string = "";
      foreach( $arr as $key => $s ) {
        if( strpos($s,"uri:") !== false ) {
          $s = str_replace("uri:","<a href=\"",$s)."\">".htmlentities(utf8_decode($s))."</a>";
          $string .= $s."; ";
        } elseif( strpos($s,"doi:") !== false ) {
          $s = "<a href=\"http://dx.doi.org/".str_replace("doi:","",utf8_decode($s))."\">".
               htmlentities(utf8_decode($s))."</a>";
          $string .= $s."; ";
        } else {
          $string .= htmlentities(utf8_decode($s))."; ";
        }
      }
    }
    return rtrim($string,"; ");
  } else {
    if( strpos($var,"uri:") !== false ) {
      $var = str_replace("uri:","<a href=\"",$var)."\">".htmlentities(utf8_decode($var))."</a>";
      return $var." ";
    } elseif( strpos($s,"doi:") !== false ) {
      return "<a href=\"http://dx.doi.org/".str_replace("doi:","",utf8_decode($s))."\">".
           htmlentities(utf8_decode($s))."</a> ";
    } else {
      return htmlentities(utf8_decode($var))." ";
    }

  }
}

/**
 * display_searchResults
 *
 * Parses and prints contents of a variable (array / SimpleXML-object). HTML-safe output.
 *
 * @param SimpleXMLObject $results XML-file containing searchresults
 * @param String $mode Print details or display text-thumbnails (possible values: 'detailed', '
     thumbnail')
```

```
92  */
93  function display_searchResults($results,$mode = "thumbnail") {
94    global $webServices,$defaultNumberOfResultsDisplayed;
95    $hilight = "dc_title";
96    $display_empty_fields = false;
97    $fieldnames = array("dc_contributor" => "Contributor(s)", "dc_coverage" => "Coverage", "dc_creator
          " => "Author(s)", "dc_date" => "Date", "dc_description" => "Description", "dc_format" => "
          Format",
98        "dc_identifier" => "Identifier", "dc_language" => "Language", "dc_publisher" => "Publisher", "
              dc_relation" => "Relation", "dc_rights" => "Rights", "dc_source" => "Source", "dc_subject
              " => "Subject", "dc_title" => "Title", "dc_type" => "Type", "repository" => "Repository")
              ;
99    if( $mode == "detailed" ) {
100     $display_fields = array("dc_contributor","dc_coverage","dc_creator","dc_date","dc_description","
            dc_format",
101       /*"dc_identifier",*/"dc_language","dc_publisher","dc_relation","dc_rights","dc_source","
            dc_subject","dc_title","dc_type","repository");
102   } else {
103     $display_fields = array("dc_title","dc_date","dc_creator","dc_type");
104   }
105   $i = 0;
106   $__output = "";
107   $we_got_entries = false;
108   $errors = "";
109   $sessions = array();
110   foreach( $results as $resultList ) {
111     $xml = simplexml_load_string(str_replace(" & "," &amp; ",$resultList));
112     $ws[] = $xml->technicalinfo->ws;
113     $offsets[] = (int)$xml->technicalinfo->offset;
114     $num_res = 0;
115     if( isset($xml->result) ) {
116       $sessions[] = $xml->technicalinfo->session;
117       $wsids[] = (int)$xml->technicalinfo->wsid;
118       // get each record:
119       foreach( $xml->result as $item ) {
120         $we_got_entries = true;
121         $num_res++;
122         $i++;
123         // print item fields:
124         if( $mode == "detailed" ) {
125           $__output .= " <div class=\"details_id\">Record Details"/* for<br />".$item->dc_identifier
                */."</div>\n";
126           $__output .= " <div class=\"item\">\n";
127           foreach( (array)$item as $key => $val ) {
128             if( in_array($key,$display_fields) && !empty($val) && isset($val) && !is_null($val) ) {
129               $val = $display_empty_fields ? $val : (array)$val;
130               $__output .= " <div class=\"row\">";
131               $description = "<span class=\"itemLabel\">".$fieldnames[$key]."</span>";
132               $css_hilight = $key == $hilight ? " hilight" : "";
133               $text = $key == "dc_creator" ? "<i>".print_var($val)."</i>" : print_var($val);
134               $__output .= $description
135                   ."<span class=\"itemText".$css_hilight."\">".$text."</span>";
136               $__output .= "</div>\n";
137             }
138           }
139           $__output .= " </div>\n";
140         } else {
141           $number = isset($_GET["offset"]) ? $i+($_GET["offset"]*count($results)) : $i;
142           $__output .= " <div class=\"number\" style=\"float:left\">".$number."</div>\n";
143           $val = (array)$item;
144           $table_class = $xml->technicalinfo->wsid % 2 == 0 ? "item_contrast" : "item";
145           $__output .= " <div class=\"".$table_class."\">\n";
146           $__output .= " <div class=\"row\">";
147           $__output .= "<span class=\"itemTextFullLine\"><a href=\"index.php?submit=details&amp;
                classid=".$xml->technicalinfo->wsid."&amp;identifier=".$item->dc_identifier."\"><b>".
```

```
                print_var($val[$display_fields[0]]).</b></a> [".trim(print_var($val[
                $display_fields[3]])).]</span>";
148           $__output .= "</div>\n";
149           $__output .= " <div class=\"row\">";
150           $__output .= "<span class=\"itemTextFullLine\">(".trim(print_var($val[$display_fields[1]])
                ).") ".print_var($val[$display_fields[2]]).</span>";
151           $__output .= "</div>\n";
152           $__output .= " </div>\n";
153         }
154         $__output .= " <div style=\"clear:both\" />\n";
155       }
156     } else {
157       $errors .= "<span style=\"color:#FF0000\">".$xml->soaperror." ".$xml."</span>\n";
158     }
159     $ws_num_res[] = $num_res;
160     $time_array[] = $xml->technicalinfo->responsetime ? (string)$xml->technicalinfo->responsetime :
          "not measured";
161   }
162   if( !$we_got_entries ) $__output .= " <div style=\"color:#FF0000;font-family:monospace\">No
        entries found for query '<b>".$_GET["terms"]."</b>'...</div>\n";
163   $j = 0;
164   foreach( $time_array as $time ) {
165     $results_txt = $ws_num_res[$j] != 1 ? "Results" : "Result";
166     $__output .= " <div style=\"text-align:right;font-family:monospace;color:#666666\">".$ws[$j]."
          Response Time: <b>".$time."</b>, ".$ws_num_res[$j]." ".$results_txt."</div>\n";
167     $j++;
168   }
169   $sessions_url = "";
170   $classIDs = "";
171   if( $sessions ) {
172     $k = 0;
173     foreach( $sessions as $session ) {
174       if( $session != "" ) {
175         $sessions_url .= $session."+";
176         $classIDs .= $wsids[$k]."+";
177       }
178       $k++;
179     }
180     $sessions_url = rtrim($sessions_url,"+");
181     $classIDs = rtrim($classIDs,"+");
182     $offset = !empty($offsets[0]) ? $offsets[0] : 0;
183     $displayed_results = (($offset*count($wsids))+1)."-".($i+($offset*count($wsids)));
184     if( $mode != "detailed" && $we_got_entries ) {
185       $query_txt = " <span style=\"font-family:monospace\">Results ".$displayed_results."</span>"
186                   ."<span style=\"font-family:monospace\"> for <b>'".$_GET["terms"]."'</b>:</span>\n"
                        ;
187       $id = 0;
188       $wsids = "";
189       foreach( $webServices as $key => $webService ) {
190         if( $_GET["ws".$key] == "true" || !isset($_GET["submit"]) ) {
191           $wsids .= "&amp;ws".$key."=true";
192         }
193         $id++;
194       }
195       $morelink = " <p style=\"clear:both\">";
196       if( $sessions_url != "" ) {
197         if( isset($_GET["offset"]) ) {
198           $morelink .= "<a href=\"javascript:history.go(-1)\" style=\"float:left\"><span style=\"
                font-weight:bold;font-family:monospace\">&lt;&lt;- Previous</span></a>";
199         }
200         $morelink .= "<a href=\"index.php?submit=moreResults".$wsids."&amp;terms=".$_GET["terms"]."&
              amp;classids=".$classIDs."&amp;sessions=".$sessions_url."&amp;offset=".($offset+
              $defaultNumberOfResultsDisplayed)."\" style=\"float:right\"><span style=\"font-weight:
              bold;font-family:monospace\">Next Results"./*((($offset*count($wsids))+1)+(
              $defaultNumberOfResultsDisplayed*count($wsids))."-".((($offset+
```

```
            $defaultNumberOfResultsDisplayed)*count($wsids))+($defaultNumberOfResultsDisplayed*count
            ($wsids))).*/" -&gt;&gt;</span></a></p><br />\n";
201       } else { $morelink .= "</p>\n"; }
202       $__output = $query_txt.$morelink.$__output." <div style=\"clear:both;\" />\n".$morelink;
203     }
204   }
205   echo $__output.$errors;
206 }
207 
208 /**
209  * display_soap_error
210  *
211  * Prints human-readable SOAP-error
212  *
213  * @param SOAPClient $soapclient Used SOAP-Client, which exposed the error
214  * @param SOAPFault $fault SOAP-Fault
215  */
216 function display_soap_error($soapclient,$fault) {
217   echo "<pre style=\"clear:both;\">\n"
218       ."<span style=\"color:#FF0000;font-weight:bold\">SOAP Fault:</span>\n"
219       ." faultcode: ( ".@$fault->faultcode." )\n"
220       ." faultstring: ( ".@$fault->faultstring." )\n"
221       ." faultactor: ( ".@$fault->faultactor." )\n"
222       ." detail: ( ".@$fault->detail." )\n"
223     ." faultname: ( ".@$fault->faultname." )\n"
224     ." headerfault: ( ".@$fault->headerfault." )</pre>\n";
225   echo "<pre><b>SOAP Request:</b>\n";
226   $lastRequest = $soapclient->__getLastRequest();
227   if( !empty($lastRequest) ) {
228     $xml = DOMDocument::loadXML($lastRequest);
229     $xml->formatOutput = true;
230     echo htmlspecialchars($xml->saveXML(), ENT_QUOTES)."\n";
231   } else {
232     echo "<i>empty</i>\n\n";
233   }
234   echo "<b>SOAP Response:</b>\n";
235   $lastResponse = $soapclient->__getLastResponse();
236   if( !empty($lastResponse) ) {
237     $xml = DOMDocument::loadXML($lastResponse);
238     $xml->formatOutput = true;
239     echo htmlspecialchars($xml->saveXML(), ENT_QUOTES)."</pre>\n";
240   } else {
241     echo "<i>empty</i>";
242   }
243   echo "</pre>\n";
244   exit;
245 }
246 
247 /**
248  * print_navigation
249  *
250  * Prints site-navigation links (start,links,about,help)
251  */
252 function print_navigation() {
253   echo "<div style=\"clear:both;width:550px;margin:0ex auto;padding:0 0 1ex 0;text-align:center\">|
        <a accesskey=\"s\" href=\"index.php\">start</a> | <a accesskey=\"l\" href=\"links.php\">links
        </a> | <a accesskey=\"a\" href=\"about.php\">about</a> | <a accesskey=\"h\" href=\"help.php
        \">help</a> |</div>\n";
254 }
255 
256 ?>
```

Quelltext C.12: xhtmlbuilders.inc.php

C.6.5 searchform.inc.php

```
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * searchform.inc.php
 * @package Diplomarbeit
 */

/**
 * XHTML-Form for searching repositories
 *
 * Including general functions
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
     works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */

if( isset($_GET["submit"]) && ($_GET["submit"] == "Search") && empty($_GET["terms"]) ) {
  $error = "Please insert a valid search term!";
  $_GET["submit"] = "";
}

if( $_GET["submit"] != "details" ) {
?>
  <div class="form">
   <div class="formHeadline">
    <a href="<?php echo $_SERVER["PHP_SELF"] ?>"><img src="includes/images/banner.jpg" style="border
        :0;border-bottom:1px solid #000000;" alt="Helmholtz Open Access" width="700px" height="50px
        " /></a>
    <p style="width:550px;margin:7px auto">Welcome to <strong>AWI Web Services</strong> for primary
        data, publications and personal portfolio.</p>
<?php
if( isset($error) ) {
  echo " <p style=\"width:550px;margin:3px auto;color:#FF0000\">".$error."</p>\n";
}
?>
   </div>
   <form style="margin-top:10px" action="<?php echo $_SERVER["PHP_SELF"] ?>" method="get">
    <div class="row" style="text-align:center;padding:0 0 1ex 0">
     <label for="terms"><span style="font-weight:bold">Search:</span></label> <input id="terms" name
         ="terms" tabindex="1" type="text" size="50" value="<?php if(isset($_GET["terms"])) {echo
         $_GET["terms"];} else {echo "insert search term here";} ?>" />
    </div>
    <div class="row" style="text-align:center;padding:0 0 0.5ex 0">
     <span style="font-size:x-small;color:#555555;">Select Repository:</span>
    </div>
    <div>
<?php
$webServices = array();
```

```
58 $id = 0;
59 foreach( $useable_webServices as $key => $webService ) {
60   echo " <div style=\"text-align:left;margin-left:295px\">\n"
61       ." <input style=\"vertical-align:middle\" id=\"ws".$key."\" name=\"ws".$key."\" type=\"
           checkbox\" value=\"true\"";
62   if( $_GET["ws".$key] == "true" || !isset($_GET["submit"]) ) {
63     echo " checked=\"checked\"";
64     $webServices[$id] = $webService;
65   }
66   echo " />\n"
67       ." <span style=\"text-align:left;font-size:x-small;vertical-align:middle;\"><label for=\"ws".
           $key."\">".$webService["name"]."</label></span>\n"
68       ." </div>\n";
69   $id++;
70 }
71 ?>
72     </div>
73     <div class="row" style="padding-top:8px;text-align:center;">
74      <input name="submit" type="submit" tabindex="2" value="Search" class="button"></input>
75     </div>
76     <div style="clear:both"></div>
77    </form>
78   </div>
79 <?php
80 }
81 ?>
```

Quelltext C.13: searchform.inc.php

C.6.6 global.css

```
1  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
2
3  /**
4   * Style sheet
5   * @package Diplomarbeit
6   */
7
8  /**
9   * Global style sheet
10  *
11  * This file defines the globally used styles for package Helmholtz_WS
12  *
13  * CSS version 2
14  *
15  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
16  * Under the following conditions:
17  * You must attribute the work in the manner specified by the author or licensor.
18  * For any reuse or distribution, you must make clear to others the license terms of this work.
19  * Any of these conditions can be waived if you get permission from the copyright holder.
20  * Your fair use and other rights are in no way affected by the above.
21  *
22  * @package Helmholtz_WS
23  * @author Bastian Onken <bonken@awi-bremerhaven.de>
24  * @copyright 2005, Bastian Onken
25  * @license http://creativecommons.org/licenses/by/2.0/
26  * @version 1.0
27  */
28
29 body {
```

```
  display:block;
  background-color:#FFFFFF;
  font-family:sans-serif;
  font-size:small;
  margin:0;
  padding:0;
  padding-top:10px;
  background-color:#EDEDED;
}
.main {
  background-color:#F4F4F4;
  border:4px solid #FFFFFF;
  padding:3px;
  width:690px;
  margin:0 auto;
}
input, textarea, select, option {
  font-size: smaller;
}
h1 {
  font-family:sans-serif;
  font-size:large;
  font-weight:bold;
  text-align:left;
  margin:0;
  padding:0px 0px 3px 0px;
}
h2 {
  font-family:sans-serif;
  font-size:medium;
  font-weight:bold;
  text-align:left;
  margin:0;
  padding:5px 0px 5px 0px;
}
p {
  margin:0;
  padding:0 0 1ex 0;
}
li {
  margin:8px 0px;
  /*background-color:#FFFFFF;*/
}
em {
  color:#006BA5;
}
a:link {
  color:#006BA5;
  font-size:small;
  text-decoration:underline;
}
a:visited {
  color:#777777;
  font-size:small;
  text-decoration:underline;
}
a:hover,a:active {
  background:#FFFFFF;
  font-size:small;
  text-decoration:underline;
}
div.formHeadline {
  text-align:center;
  margin:0;
  padding:0;
}
```

```
96  div.searchResults {
97    width:550px;
98    margin:0 auto;
99  }
100 #footer {
101   clear:both;
102   width:550px;
103   margin:1ex auto;
104   padding:1ex 0;
105   text-align:center;
106 }
107 .number {
108   width:19px;
109   height:17px;
110   font-family:monospace;
111   font-size:small;
112   font-weight:bold;
113   background-color:#003366;
114   color:#FFFFFF;
115   border:1px solid #FFFFFF;
116   vertical-align:middle;
117   text-align:center;
118   margin:1ex 0;
119   float:left;
120 }
121 .details_id {
122   padding:2px 5px;
123   font-family:monospace;
124   font-size:small;
125   font-weight:bold;
126   background-color:#003366;
127   color:#FFFFFF;
128   border:1px solid #FFFFFF;
129   vertical-align:middle;
130   text-align:center;
131   margin:1ex 0;
132   float:left;
133 }
134 .item {
135   float:right;
136   width:520px;
137   border:1px solid #FFFFFF;
138   margin:0 0 1ex 0;
139   padding-bottom:1ex;
140   background-color:#F4F4F4;
141   font-size:12px;
142 }
143 .item_contrast {
144   float:right;
145   width:520px;
146   border:1px solid #FFFFFF;
147   margin:0 0 1ex 0;
148   padding-bottom:1ex;
149   background-color:#F4F4F4;
150   font-size:12px;
151 }
152 .hilight {
153   font-weight:bold;
154   font-size:small;
155 }
156 input.button {
157   font-size:x-small;
158   font-weight:bold;
159   color:#FFFFFF;
160   background:#003366;
161   border-top: 1px #CCC solid;
```

```
162   border-right: 1px #000 solid;
163   border-bottom: 1px #333 solid;
164   border-left: 1px #999 solid;
165   padding:2px 10px;
166 }
167 input.button:hover {
168   border-top: 1px #333 solid;
169   border-right: 1px #999 solid;
170   border-bottom: 1px #CCC solid;
171   border-left: 1px #000 solid;
172 }
173 div.form {
174   width:700px;
175   font-size:small;
176   padding:0px 0px 10px 0px;
177   border:1px solid #666666;
178   border-top:1px solid #000000;
179   margin:10px auto 1ex auto;
180   background-color:#F5F5F5;
181 }
182 div.row {
183   clear: both;
184   padding-top: 5px;
185 }
186 div.row span.formLabel {
187   color: #006BA5;
188   font-weight: bold;
189   float: left;
190   width:165px;
191   text-align: right;
192 }
193 div.row span.formField {
194   width: 260px;
195   float: right;
196   text-align: left;
197 }
198 div.row span.itemLabel {
199   color:#666666;
200   font-weight:bold;
201   float:left;
202   width:80px;
203   text-align:left;
204   vertical-align:top;
205   padding:0px 3px;
206 }
207 div.row span.itemText {
208   width:420px;
209   float:right;
210   text-align:left;
211   padding:0px 3px 0px 0px;
212   overflow:hidden;
213 }
214 div.row span.itemTextFullLine {
215   width:510px;
216   float:right;
217   text-align:left;
218   padding:0px 3px 0px 0px;
219   overflow:hidden;
220 }
```

Quelltext C.14: global.css

C.6.7 index.php

```php
<?php

/* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */

/**
 * index.php
 * @package Diplomarbeit
 */

/**
 * Startpage
 *
 * Displays the initial XHTML-page containing form and possibly searchresults
 *
 * PHP version 5
 *
 * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative works; to make commercial use of the work
 * Under the following conditions:
 * You must attribute the work in the manner specified by the author or licensor.
 * For any reuse or distribution, you must make clear to others the license terms of this work.
 * Any of these conditions can be waived if you get permission from the copyright holder.
 * Your fair use and other rights are in no way affected by the above.
 *
 * @package Helmholtz_WS
 * @author Bastian Onken <bonken@awi-bremerhaven.de>
 * @copyright 2005, Bastian Onken
 * @license http://creativecommons.org/licenses/by/2.0/
 * @version 1.0
 */

require_once("includes/header.inc.php");
require_once("includes/searchform.inc.php");

if( isset($_GET["submit"]) && $_GET["submit"] == "Search" ) {
  echo " <div class=\"searchResults\">\n";
  $sessions = array();
  $time_response_start = microtime("float");
  display_searchResults(multiSearch("simpleSearch",$_GET["terms"],null,$sessions,array_keys($webServices),0),"thumbnails");
  $time_response_end = microtime("float");
  $time_response = $time_response_end - $time_response_start;
  echo " <div style=\"width:300;text-align:right;clear:both;font-family:monospace\">Script Time: <b>".number_format($time_response,3)."s</b></div>\n";
  echo " </div>\n";
}

if( isset($_GET["submit"]) && $_GET["submit"] == "moreResults" ) {
  echo " <div class=\"searchResults\">\n";
  $sessions = explode(" ",$_GET["sessions"]);
  $classIDs = explode(" ",$_GET["classids"]);
  $time_response_start = microtime("float");
  display_searchResults(multiSearch("moreResults",$_GET["terms"],null,$sessions,$classIDs,$_GET["offset"]),"thumbnails");
  $time_response_end = microtime("float");
  $time_response = $time_response_end - $time_response_start;
  echo " <div style=\"width:300;text-align:right;clear:both;font-family:monospace\">Script Time: <b>".number_format($time_response,3)."s</b></div>\n";
  echo " </div>\n";
}

if( isset($_GET["submit"]) && ($_GET["submit"] == "details") && isset($_GET["classid"]) ) {
  echo " <div style=\"width:702px;margin:10px auto 20px auto\">\n"
```

```
59      ." <a href=\"index.php\"><img src=\"includes/images/banner.jpg\" style=\"border:1px solid
            #000000\" alt=\"Helmholtz Open Access\" /></a>\n"
60      ." </div>\n";
61    echo " <div class=\"searchResults\">\n";
62    $class = $useable_webServices[$_GET["classid"]]["class"];
63    $name = $useable_webServices[$_GET["classid"]]["name"];
64    $wsdl = $useable_webServices[$_GET["classid"]]["wsdl"];
65    $ws = new $class($name,$wsdl);
66    $time_response_start = microtime("float");
67    $xml = $ws->detailsForObject($_GET["identifier"]);
68    $time_response_end = microtime("float");
69    $time_response = $time_response_end - $time_response_start;
70    if( strpos($xml,"</search>") !== false ) {
71      //String was found
72      $output = str_replace("</search>","<technicalinfo><ws>".$_GET["ws"]."</ws><responsetime>".
            number_format($time_response,3)."s</responsetime></technicalinfo></search>",$xml);
73    } else {
74      $output = str_replace("<search/>","<search><technicalinfo><ws>".$_GET["ws"]."</ws><responsetime>
            ".number_format($time_response,3)."s</responsetime></technicalinfo></search>",$xml);
75    }
76    display_searchResults((array)$output,"detailed");
77    echo " </div>\n";
78  }
79
80  require_once("includes/footer.inc.php");
81
82  ?>
```

Quelltext C.15: index.php

C.6.8 help.php

```
1   <?php
2
3   /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5   /**
6    * help.php
7    * @package Diplomarbeit
8    */
9
10  /**
11   * XHTML-page 'Help'
12   *
13   * Shows helppage
14   *
15   * PHP version 5
16   *
17   * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
         works; to make commercial use of the work
18   * Under the following conditions:
19   * You must attribute the work in the manner specified by the author or licensor.
20   * For any reuse or distribution, you must make clear to others the license terms of this work.
21   * Any of these conditions can be waived if you get permission from the copyright holder.
22   * Your fair use and other rights are in no way affected by the above.
23   *
24   * @package Helmholtz_WS
25   * @author Bastian Onken <bonken@awi-bremerhaven.de>
26   * @copyright 2005, Bastian Onken
27   * @license http://creativecommons.org/licenses/by/2.0/
28   * @version 1.0
```

```
29  */
30
31  require_once("includes/header.inc.php");
32
33  print_navigation();
34
35  ?>
36      <div class="main">
37       <h1>Help</h1>
38       <h2>Search</h2>
39       <p>To search for entries in the offered systems, just type your keyword(s) in the blank field
             an click on the <em>Search</em>-button.</p>
40       <p>Various search methods are supported (especially the Pangaea-based part).</p>
41       <ol>
42        <li>The search is <em>not</em> case sensitive<br /><br />
43         Searching for <em>neumayer</em> returns the same results as the term <em>NEUMAYER</em> or <em
               >nEuMaYeR</em>
44        </li>
45        <li>It works with the <em>AND</em> logic by default while searching for more keywords<br /><br
               />
46         You can use <em>OR</em> if you want either one or another keyword (or both)<br />
47         This function is not supported by Fedora (yet)
48        </li>
49        <li>With the word variation / stemming-technology the MISAWIsta will not search only for your
             search terms, but also for words that are similar to some or all of those terms<br /><br
             />
50         The &quot;+&quot;-symbol disable stemming technology and make the search case sensitive (e.g.
               <em>+neumayer</em> gets no results, but <em>+Neumayer</em> do)<br />
51         The &quot;-&quot;-symbol exclude a word from your search, which may not contain in the search
               result<br />
52         The &quot;&sim;&quot;-symbol search for variants in spelling (e.g. <em>&sim;k&ouml;nig</em>
             finds also <em>karig</em>)<br />
53         This function is not supported by Fedora (yet)
54        </li>
55        <li>Also available are wildcards, which allows to substitute unknown characters for part of
             the item for which you are searching<br /><br />
56         <strong>?</strong> : specifies one alphanumeric character (e.g. <em>k?nig</em> gets <em>k&
             ouml;nig</em> or <em>kanig</em> and so on)<br />
57         <strong>*</strong> : specifies zero or more of any alphanumeric character (e.g. <em>meteor*</
             em> gets <em>meteor</em>, <em>meteorology</em>, <em>meteoroid</em> et cetera)<br />
58         <strong>[]</strong> : specifies any single character in a set (e.g. <em>gr[ou]be</em> search
             for <em>grobe</em> and <em>grube</em>)<br />
59         <strong>{}</strong> : specifies one of each pattern separated by a comma (e.g. <em>meteor{
             ology, ological, oid}</em> gets <em>meteorology</em>, <em>meteorological</em> and <em>
             meteoroid</em>)<br />
60         <strong>^</strong> : specifies one of any charcter not inculded in a set (e.g. st[^oa]ck
             excludes <em>stock</em> and <em>stack</em>, but locates <em>stick</em> and <em>stuck</em
             >)<br />
61         <strong>-</strong> : specifies a range of characters in a set (e.g. <em>c[a-r]t</em> includes
               every three-letter word from <em>cat</em> to <em>crt</em>)<br />
62         This function is not supported by Fedora (yet)
63        </li>
64        <li>A search term can also contains phrases<br /><br />
65         Set the words in double quotes or join them with the &quot;-&quot;-symbol. (e.g. <em>"phrase
               one"</em> or <em>phrase-one</em>)<br />
66         This function is not supported by Fedora (yet)
67        </li>
68        <li>Pangaea supports the search in specific fields (e.g. <em>author:grobe</em> or <em>date
             :2004</em> and so on) <br /><br />
69         <em>project:</em> search for keywords in projects<br />
70         <em>projectlabel:</em> matches a project label<br />
71         <em>author:</em> search for authors of datasets or assigned references<br />
72         <em>citation:author:</em> search for authors of datasets only in the citation<br />
73         <em>pi:</em> search for datasets with Principal Investigator (PI)<br />
74         <em>citation:</em> search for keywords in the citation<br />
75         <em>reference:</em> search for keywords in assigned references<br />
```

```
76        <em>date:</em> search for datasets or assigned references published in a specific year<br />
77        <em>parametername:</em> search for keywords in parameter names<br />
78        <em>methodname:</em> search for keywords in method names<br />
79        <em>eventlabel:</em> search for event labels
80       </li>
81      </ol>
82     </div>
83 <?php
84
85 require_once("includes/footer.inc.php");
86
87 ?>
```

Quelltext C.16: help.php

C.6.9 links.php

```
1  <?php
2
3  /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5  /**
6   * links.php
7   * @package Diplomarbeit
8   */
9
10 /**
11  * XHTML-page 'Links'
12  *
13  * Shows related Links
14  *
15  * PHP version 5
16  *
17  * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
        works; to make commercial use of the work
18  * Under the following conditions:
19  * You must attribute the work in the manner specified by the author or licensor.
20  * For any reuse or distribution, you must make clear to others the license terms of this work.
21  * Any of these conditions can be waived if you get permission from the copyright holder.
22  * Your fair use and other rights are in no way affected by the above.
23  *
24  * @package Helmholtz_WS
25  * @author Bastian Onken <bonken@awi-bremerhaven.de>
26  * @copyright 2005, Bastian Onken
27  * @license http://creativecommons.org/licenses/by/2.0/
28  * @version 1.0
29  */
30
31 require_once("includes/header.inc.php");
32
33 ?>
34     <div class="main">
35      <h1>Links</h1>
36      <h2>Alfred Wegener Institute for polar and marine research</h1>
37      <p style="text-align:justify">Polar and Marine research are central themes of Global system and
            Environmental Science. The Alfred Wegener Institute conducts research in the Arctic, the
          Antarctic and at temperate latitudes. It coordinates Polar research in Germany and
          provides both the necessary equipment and the essential logistic back up for polar
          expeditions. Recent additional research themes include North Sea Research, contributions
          to Marine Biological Monitoring, Marine Pollution Research, Investigation of naturally
          occuring marine substances and technical marine developments.</p>
```

```
38      <p style="text-align:right"><a href="http://www.awi-bremerhaven.de"><img src="http://www.awi-
            bremerhaven.de/Pics/all_awilogo-d.gif" alt="Logo AWI Bremerhaven" /></a></p>
39      <h2>Pangaea</h1>
40      <p style="text-align:justify">PANGAEA is a public digital library for science aimed at
            archiving, publishing and distributing georeferenced data with special emphasis on
            environmental, marine and geological basic research.</p>
41      <p style="text-align:right"><a href="http://www.pangaea.de"><img src="http://www.pangaea.de/
            PICS/PanLogo.gif" alt="Logo Pangaea" /></a></p>
42     </div>
43  <?php
44
45  require_once("includes/footer.inc.php");
46
47  ?>
```

Quelltext C.17: links.php

C.6.10 about.php

```
1   <?php
2
3   /* vim: set expandtab tabstop=4 shiftwidth=4 softtabstop=4: */
4
5   /**
6    * about.php
7    * @package Diplomarbeit
8    */
9
10  /**
11   * XHTML-page 'About'
12   *
13   * Shows responsible persons
14   *
15   * PHP version 5
16   *
17   * LICENSE: You are free: to copy, distribute, display, and perform the work; to make derivative
         works; to make commercial use of the work
18   * Under the following conditions:
19   * You must attribute the work in the manner specified by the author or licensor.
20   * For any reuse or distribution, you must make clear to others the license terms of this work.
21   * Any of these conditions can be waived if you get permission from the copyright holder.
22   * Your fair use and other rights are in no way affected by the above.
23   *
24   * @package Helmholtz_WS
25   * @author Bastian Onken <bonken@awi-bremerhaven.de>
26   * @copyright 2005, Bastian Onken
27   * @license http://creativecommons.org/licenses/by/2.0/
28   * @version 1.0
29   */
30
31  require_once("includes/header.inc.php");
32
33  ?>
34      <div class="main">
35       <p>Responsible for this portal are:</p>
36       <h2>Bastian Onken (developer)</h2>
37       <p style="text-align:left">
38        Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung<br />
39        in der Helmholtz-Gemeinschaft<br />
40        Am Handelshafen 12<br />
41        27570 Bremerhaven<br />
```

```
42      Tel.: +49 (0)471 4831-1781<br />
43      Fax: +49 (0)471 4831-1590<br />
44      Email: <a href="mailto:bonken@awi-bremerhaven.de">bonken@awi-bremerhaven.de</a><br />
45      Internet: <a href="http://www.awi-bremerhaven.de/People/show?bonken">Personal homepage of
            Bastian Onken</a>
46     </p>
47     <h2>Dr. Ana Macario (webmaster)</h2>
48     <p style="text-align:left">
49      Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung<br />
50      in der Helmholtz-Gemeinschaft<br />
51      Am Handelshafen 12<br />
52      27570 Bremerhaven<br />
53      Tel.: +49 (0)471 4831-1435<br />
54      Fax: +49 (0)471 4831-1149<br />
55      Email: <a href="mailto:amacario@awi-bremerhaven.de">amacario@awi-bremerhaven.de</a><br />
56      Internet: <a href="http://www.awi-bremerhaven.de/People/show?amacario">Personal homepage of Dr
            . Ana Macario</a>
57     </p>
58    </div>
59 <?php
60
61 require_once("includes/footer.inc.php");
62
63 ?>
```

Quelltext C.18: about.php

Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Stotel, den 30. September 2005 _____