Hochschule Bremerhaven
An der Karlstadt 8
27568 Bremerhaven

Sommersemester 2005

— Diplomarbeit —
zur Erlangung des Grades eines
Diplom-Wirtschaftsinformatikers (FH)

Konzeptionierung und Prototypenentwicklung eines auf der Web Services-Architektur basierenden Portals zur Suche von Publikationen und Daten der Meteorologie in einer wissenschaftlichen Großforschungseinrichtung

Benny Bräuer
Matr.-Nr. 22342

Betreut durch Prof. Dr. Dieter Viefhues (Referent)
und Dr. Ana Macario (Korreferentin)

Vorgelegt von: Benny Bräuer
Matr.-Nr.: 22342
Gattje 3
27632 Midlum
+49(4741)1814 – 00

Version vom: 29. September 2005

Referent: Prof. Dr. Dieter Viefhues
Hochschule Bremerhaven
An der Karlstadt 8
27568 Bremerhaven
+49(471)4823 – 448

Korreferentin: Dr. Ana Macario
Alfred-Wegener-Institut
für Polar- und Meeresforschung
Am Handelshafen 12
27570 Bremerhaven
+49(471)4831 – 1435

Gesetzt mit LaTeX 2_ε

Hebt man den Blick,
so sieht man keine Grenzen.
(aus Japan)

Danksagung
Mein Dank gilt allen, die mir während des Studiums und der Diplomarbeit mit Rat und Tat, Freundschaft und Fürsorge, Lob und Kritik treu zur Seite standen.
Danke!

Zusammenfassung

Es wurde ein Suchportal (MISAWIsta) erschaffen, mit dem Interessenten nach Daten der Meteorologie suchen können. Dabei stehen neben einer einfachen textfeldbasierten Suche auch spezielle Suchfunktionen für Forschungsschiffe und Stationen zur Verfügung. Es besteht die Möglichkeit sich die Ergebnisse, sofern es sich dabei um Messdaten handelt, anzuzeigen und als CSV- oder MarineXML-Datei herunterzuladen. Zur Realisierung wird auf die Web Service-Technologie gesetzt.

Das Portal fungiert als dynamischer SOAP-Client auf PHP-Basis. Es werden Verbindungen zu Pangaea (Messdaten) und dem Fedorasystem (Publikationen des Alfred-Wegener-Instituts) hergestellt und diese Systeme durchsucht. Die zurückgelieferten Ergebnisse werden aufbereitet und dem Benutzer präsentiert. Diese Arbeit beschreibt die Konzeptionierung sowie die Entwicklung des Prototypen.

Abstract

A portal named MISAWIsta (Meteorological Information System) was created which allows users to search for primary data and related publications found in Pangaea and Fedora information systems respectively. The user has the possibility (in case of measurement data) to view the results and download them as CSV- or MarineXML-file.

Given that Pangaea and Fedora information systems have been implemented as a Web Service, a PHP-based SOAP-Client was written to expose these data. The SOAP-Client prepares the query results and presents them to the user. This work describes the conception and development of the prototype.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1 Einleitung

1.1 Motivation

Jahr für Jahr liefern Wissenschaftler weltweit eine schier endlose Menge an Messergebnissen, Berechnungen und anderen Forschungsdaten. Doch während es für die Forscher kein Problem darstellt, Daten en masse zu erzeugen, haben die Fachkräfte der Informationstechnologie alle Hände voll zu tun, diese Daten zu speichern, zu verwalten und nutzbar aufzubereiten.

Mit fortschreitender Entwicklung der IT stellten diese Aufgaben aber das eher kleinere Problem dar. In den Vordergrund rückte nunmehr die *sinnvolle* Speicherung, Verwaltung und Aufbereitung.

Für die Rechenzentren der Forschungsinstitute kamen nun Fragen auf wie z.B.: „Wie stellen wir die Daten der Allgemeinheit zur Verfügung?“, „Wie können wir Ressourcen teilen oder miteinander kombinieren?“, „Speichern wir Zentral in einer großen Datenbank / einem Data-Warehouse oder lassen wir die bisherige Verteilung bestehen?“ oder, im Anbetracht moderner Kommunikationstechnologien im Netz „Wie kann ich mittels weltweit gültiger Standards meine Daten zur Verfügung stellen oder mit anderen teilen ohne meine bestehende Struktur großartig zu verändern?“.

In diesem Fall geht es um die Daten des Alfred-Wegener-Instituts für Polar- und Meeresforschung in Bremerhaven, im Speziellen um dessen meteorologische Daten. Bei den Daten handelt es sich zum Einen um Publikationen jedweder Form und zum Anderen um Messergebnisse und -aufzeichnungen. Alle Daten lagern in zwei verschiedenen Systemen, welche unabhängig voneinander agieren.

Aufgabe soll sein, diese Systeme nach entsprechenden Kriterien zu durchsuchen und die Ergebnisse darzustellen. Das Mittel, um das gewünschte Resultat zu bekommen, heißt „Web Services via SOAP“ und wird ein wichtiger Bestandteil dieser Arbeit.

1.2 Das Alfred-Wegener-Institut

Träger dieser Diplomarbeit ist das Alfred-Wegener-Institut (AWI) für Polar- und Meeresforschung in Bremerhaven. Im Jahre 1980 wurde das Institut als Stiftung des öffentlichen Rechts gegründet.

Das Institut ist eines von 15 Forschungszentren, welche zu den Hermann von Helmholtz-Gemeinschaft Deutscher Forschungszentren (HGF) gehören. Die Finanzierung erfolgt zu 90% aus Geldern des Bundesministerium für Bildung und Forschung (BMBF). Die restlichen Mittel werden von den Ländern Bremen (8%) sowie Brandenburg und Schleswig-Holstein (jeweils 1%) gestellt. Im Jahre 2005 verfügte das AWI über ein Etat von ca. 100 Mio. € und beschäftigt rund 780 Mitarbeiter.

Die Stiftung führt wissenschaftliche Projekte in der Arktis, Antarktis und den gemäßigten Breiten durch und koordiniert die Polarforschung in Deutschland. Des Weiteren stellt sie 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.[1]

Die Stiftung umfasst:

- das Alfred-Wegener-Institut für Polar- und Meeresforschung Bremerhaven,
- die Forschungsstelle Potsdam,
- die Biologische Anstalt Helgoland sowie
- die Wattenmeerstation Sylt

An der Spitze der Stiftung stehen das Direktorium, der wissenschaftliche Beirat sowie ein Kuratorium. Darauf folgen in der Struktur auf der einen Seite die wissenschaftlichen Fachbereiche, Technologien und allgemeine Dienste. Auf der anderen Seite steht das Forschungsprogramm MarCoPolI (**Mar**ine, **Co**ast, **Pol**ar, **I**nfrastruktur), vgl. Abbildung 1.1.

[1] vgl. [AWI05 a]

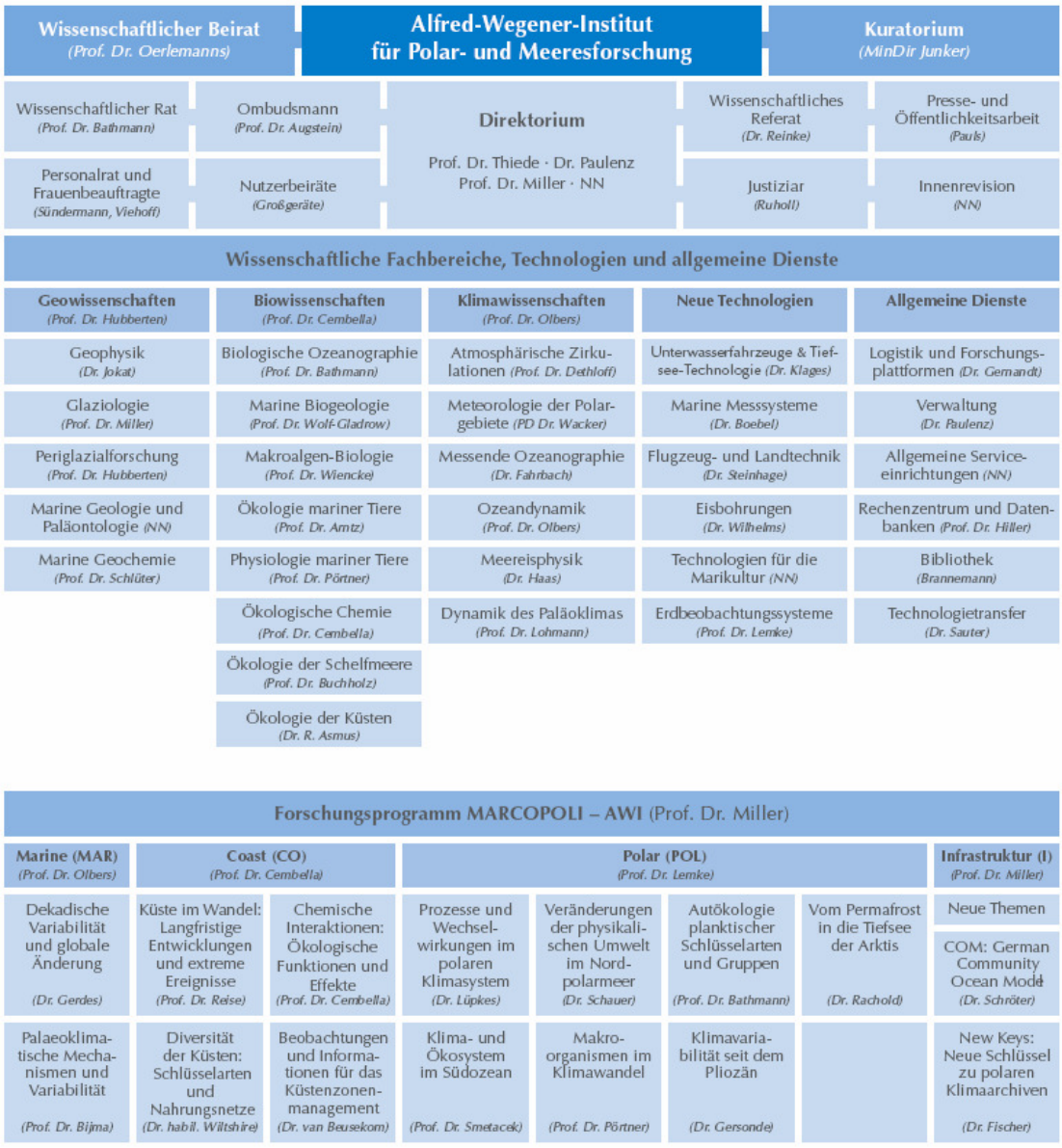


Abbildung 1.1: Organigramm der Stiftung Alfred-Wegener-Institut, s. [AWI05 b]

1.3 Aufgabenstellung

Aufgabe der vorliegenden Diplomarbeit ist die Konzeptionierung und Prototypenentwicklung eines Portals zur Suche nach meteorologischen Publikationen und Daten. Das Portal[2] soll neben der einfachen Suche mittels Eingabe in einem Text-

[2] s. Punkt 2.4.4.1 auf Seite 27

feld[3] eine spezialisierte Suche, zugeschnitten auf die Meteorologie, enthalten. Diese Suche soll die Möglichkeit bieten, sich Daten nach bestimmten Kriterien, wie z.B. eine Station oder eine Schiffsreise in Verbindung mit einer Messmethode, ausgeben zu lassen. Ebenfalls notwendig ist eine detailliertere Ansicht der zurückgelieferten Suchergebnisse zur genaueren Begutachtung sowie die Möglichkeit sich die Messdaten zur weiteren Verwendung herunterzuladen.

Der wichtigste Bestandteil ist aber nicht die Suche und Darstellung, sondern die Realisierung der internen Arbeit mittels Web Service-Technologie. Das Portal dient dabei als Client für mehrere, unabhängig voneinander arbeitende Web Services.

1.4 Ziel der Arbeit

Ziel der Diplomarbeit ist die Umsetzung der Aufgabenstellung. Durch die Web Services-Technologie soll gewährleistet werden, dass neben den bereits verwendeten Datenarchiven und Informationssystemen auch neue hinzugefügt werden können, sofern sie kompatibel zu Web Services sind bzw. diese bereitstellen.

Ferner ist zu erläutern, welchen Nutzen man von dieser Technologie erhält. Da es sich bei der Entwicklung um einen Prototypen handelt, welcher eine Art *funktionstätiges Grundgerüst* für die genannte Aufgabe darstellt, muss ebenfalls beschrieben werden, wie das Portal bzw. gesamte Programm erweitert und ggf. optimiert werden kann.

1.5 Aufbau der Arbeit

Nach der Einleitung, welche neben der Motivation auch die Aufgabenstellung und das Ziel der Diplomarbeit enthält, folgt das Kapitel Grundlagen. In diesem werden grundlegende Bestandteile der Theorie und Praxis der Arbeit, also z.B. verwendete Daten oder Programmiersprachen, erläutert.

Kapitel Drei stellt die Anforderungen an das System dar und schildert die Analyse des bestehenden Systems. Kapitel Vier befasst sich mit einer wichtigen Vorbereitung der Entwicklung, nämlich dem Transport vieler meteorologischer Daten von einer Arbeitsdatenbank in ein Web Service-fähiges Datenarchiv.

Um die Entwicklung des Portals geht es in Kapitel Fünf, das weiterhin auch die Implementierung sowie die Demonstration der fertigen Entwicklung enthält. Kapitel Sechs widmet sich den Aussichten und in Kapitel Sieben wird ein abschließendes

[3] ähnlich der bekannten Suchmaschine Google (http://www.google.de)

Fazit gezogen. Darauf folgt der Anhang mit dem Abkürzungs- und Literaturverzeichnis sowie den Quelltexten.

Eine Diplomarbeit ist als ein wichtiges Projekt zu betrachten. Von daher gilt es, sich an gewisse Regeln des Projektmanagement zu halten. Eine dieser Regeln ist eine klare, zeitliche Abgrenzung des Projekts. Diese Abgrenzung ist in Abb. 1.2 dargestellt. Die darin enthaltenen Zeitangaben sind angestrebt, können sich aber im Verlaufe des Projekts verschieben.

Der Projektbeginn ist der 1. Juli 2005. Dieser Monat dient der Vorbereitung auf das eigentliche Thema. Dazu gehören neben einer fachlichen Auseinandersetzung mit den benötigten Daten, Systemen und Anwendungen auch die Treffen mit den beteiligten Wissenschaftlern zwecks Diskussion der Datenaufbereitung. Da es sich hierbei im weitesten Sinne um „politische“ Entscheidungen handelt (deren Umsetzung bei einem Institut dieser Größe eine gewisse Zeit in Anspruch nehmen kann) erfolgt die Anmeldung erst Ende Juli / Anfang August, um eine als negativ erachtete Ausdehnung des Projektzeitraums durch nicht-beeinflussbare Entscheidungsprozesse zu vermeiden. Die Entwicklung des Portals sowie die Dokumentation erfolgen dann hauptsächlich im August / September. Der Oktober dient in erster Linie der Begutachtung durch die Referenten sowie der Vorbereitung bzw. Durchführung des Kolloquiums. Falls notwendig dient er auch als Puffer für unvorhersehbare Verzögerungen.

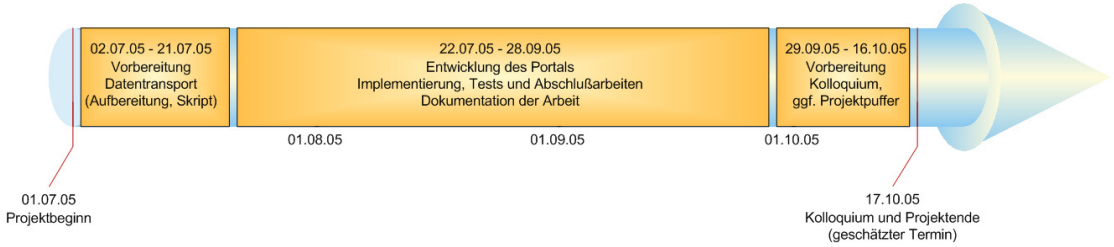


Abbildung 1.2: Zeitliche Abgrenzung des Projekts

2 Grundlagen

2.1 Daten

Da es sich beim dem Portal um Datenverarbeitung handelt, muss zunächst geklärt werden, mit welchen Arten von Daten gearbeitet wird. Die folgenden Seiten geben Aufschluss über diese Frage. Erklärt werden die Metadaten sowie die meteorologischen Daten, im Anschluss die *kennzeichnenden Schlüssel* der Daten.

2.1.1 Metadaten

> „Metadaten sind maschinenlesbare Informationen über elektronische Ressourcen oder andere Dinge.“ – Sir Timothy John Berners-Lee, „Erfinder“ des World Wide Web und Direktor des W3 Consortium

Metadaten sind strukturierte Daten, welche Informationen über andere Daten bzw. Informationsressourcen enthalten. Durch die Metadatenbeschreibung werden eben diese Informationsressourcen leichter auffindbar gemacht. Beschrieben werden können z.B. Bücher, Dateien oder Datenbanken. Metadaten liefern also Grundinformationen über ein Dokument, wie z.B. Angaben über Titel, Autor oder Zeitpunkt der Veröffentlichung. Es existiert aber keine allgemeingültige Unterscheidung zwischen Metadaten und normalen Daten. Das Prinzip der Metadaten ist auch keine Neuerrungenschaft, sondern findet bspw. in Bibliotheken seit Jahrhunderten Anwendung (u.a. in Karteisystemen, in welchen neben Titel, Autor auch der Standort des Buches verzeichnet ist).

Um diese Datenart nun effektiv und sinnvoll nutzen zu können, ist ein gewisser Standardisierungsgrad vorausgesetzt. Daher werden stets neue Ansätze in der Ressourcenbeschreibung und den entsprechenden Verfahren der Informationsvermittlung gesucht, welche in elektronischen Netzen auf einen effizienten und kostengünstigen/kostenlosen Einsatz optimiert sind. Ein solcher Ansatz ist die *Dublin Core Metadata Initiative*.

2.1.1.1 Dublin Core Metadata Initiative

Dublin Core ist ein standardisiertes, weltweit gültiges Format für Metadaten zur Beschreibung von Dokumenten und anderen Objekten im Internet. Die Dublin Core Metadata Initiative[1] gehört zu den bekanntesten und in der internationalen Diskussion wichtigsten Entwicklungen, welche sich dem Konzept „Metadaten zur Erschließung digitaler Ressourcen“ widmen.

1995 haben sich Bibliothekare, Informationswissenschaftler und Informatiker in Dublin, Ohio (daher der Name) zusammengetan, um dieses Konzept zu realisieren. Entwickelt wurde ein Kernsatz (Core) von 15 Elementen, auch Dublin Core Metadata Element Set genannt, deren Einfachheit es jedem ermöglichen, seine Dokumente mit Metadaten zu versehen. Durch die Standardisierung wird Dublin Core inzwischen auch von den meisten Suchmaschinen im Internet unterstützt. Die Darstellung kann bspw. in XML/RDF (Resource Description Framework) erfolgen. Es existieren keine genauen Regeln für die Belegung der einzelnen Felder, daher sollte Dublin Core als eine Art „kleinster gemeinsamer Nenner“ für den Austausch von Metadaten betrachtet werden. Alle Felder sind optional und können ggf. auch mehrfach vorkommen. Die Merkmale von Dublin Core sind[2]:

- Einfachheit
  - kann sowohl von Experten als auch von Laien eingesetzt werden
  - Elemente besitzen eine allgemeinverständliche Semantik
- semantische Interoperationalität
  - kann durch universellen Charakter der Deskriptoren in den verschiedensten Fachrichtungen eingesetzt werden
  - verbessert die interdisziplinäre Suche nach Informations-Ressourcen
- internationale Verbreitung
  - profitiert von der Beteiligung internationaler Organisationen und Experten, welche die Verbreitung dieses Standards vorantreiben
- Erweiterbarkeit
  - bietet eine ökonomische Alternative zu den komplexeren Metadaten-Modellen
  - mit Hilfe von Sub-Elementen flexibel erweiterbar

[1] vgl. http://dublincore.org/
[2] vgl. [CELab05]

Die 15 Dublin Core Metadaten-Elemente lauten wie folgt:

Element	Beschreibung
Title	Titel des Objekts
Creator	Erzeuger
Subject	inhaltliche Bezüge, Schlagwörter
Description	Beschreibung
Publisher	Herausgeber
Contributor	Beitragende
Date	Datum
Type	Objekttyp
Format	Format
Identifier	eindeutige Bezeichnung
Source	Quellen- bzw. Originalangabe für nichtoriginäre Beiträge
Language	Sprache
Relation	Beziehung zu anderen Objekten
Coverage	räumliche und zeitliche Zuordnung des Objekts
Rights	Urheberrechte / Nutzungsbedingungen

Tabelle 2.1: Dublin Core Metadaten-Elemente

2.1.1.2 Weitere Formate

Neben dem Dublin-Core-Schema gibt es noch eine Vielzahl von weiteren Metadatenformaten, z.B. den nach ISO 19115 genormten Standard für Metainformationen. Diese Norm beinhaltet neben den Angaben für die Identifikation, Qualität etc. auch Informationen zum räumlichen und zeitlichen Schema der Daten sowie zum Referenzsystem. Dadurch kann eine Georeferenzierung der Publikationen erfolgen. Insgesamt enthält die Norm Definitionen für 409 Metadatenelemente, von denen ca. 20 verpflichtende Angaben sind.

Daneben existieren im Bereich der erdsystemnahen Wissenschaften noch andere Standards, wie das CIP (Catalogue Interoperability Protocol[3]), der DIF-Standard (Directory Interchange Format des Global Change Master Directory[4]) oder dem *Content Standard for Digital Geospatial Metadata* des *Federal Geographic Data Committee* (FGDC)[5].

Diese Formate kommen in dieser Arbeit nicht zum Einsatz sondern seien der Vollständigkeit halber am Rande erwähnt. Es ist aber nicht auszuschließen, dass in Zukunft im Rahmen einer Portalerweiterung mit ihnen gearbeitet wird.

[3]vgl. http://www.loc.gov/z3950/agency/profiles/cip.html
[4]vgl. http://gcmd.nasa.gov
[5]vgl. http://www.fgdc.gov/standards/standards.html

2.1.2 Meteorologische Daten

Unter Meteorologie versteht man ein Teilgebiet der Physik, des Öfteren auch der Geowissenschaften, welches sich unter anderem auf die Dynamik der unteren Erdatmosphäre und dem dadurch hervorgerufenen Wetter konzentriert. Weiterhin zugehörig ist die Beobachtung von klima- und wetterrelevanten Vorgängen in der Atmosphäre vom Boden, aus der Luft und aus dem Weltraum (Klimatologie). Das heutige Verständnis der Meteorologie ist vom Leitbegriff der „Physik der Atmosphären“ geprägt.[6]

Bei dieser Wissenschaft fallen wie eingangs erwähnt riesige Mengen von Daten an, welche von den unterschiedlichsten Geräten (über Thermographen bis hin zum Wettersatelliten) aufgezeichnet werden. Diese werden nun, nach mehr oder minder komplexen Methoden, in Arbeits- oder Archivdatenbanken importiert, von denen aus sie für Abruf und Weiterverarbeitung bereitstehen.

SQL Advantage - [Result 1 From Session 1]

File Edit View Server Query Window Help

```
select * from ObseDat where DatumUhrzeit > "2005"
```

	DatumUhrzeit	Wolkenuntergrenze	HorSicht	Windrichtung	Windgeschw	Temperatur	Taupunkt
1	Jan 1 2005 3:00AM	4	97	170	3.0	-3.8999999999999999	-5.9000000000000004
2	Jan 2 2005 12:00AM	8	97	50	2.0	-3.1000000000000001	-7.0
3	Jan 2 2005 3:00AM	8	97	30	3.0	-3.8999999999999999	-6.2000000000000002
4	Jan 2 2005 6:00PM	4	97	30	5.0	-1.0	-3.3999999999999999
5	Jan 2 2005 9:00PM	4	97	30	6.0	-0.90000000000000002	-3.6000000000000001
6	Jan 3 2005 12:00AM	NULL	97	30	5.0	-1.5	-3.2999999999999998
7	Jan 3 2005 3:00AM	3	97	40	6.0	-3.2999999999999998	-4.0999999999999996
8	Jan 3 2005 9:00PM	1	97	30	10.0	-0.20000000000000001	-0.5
9	Jan 4 2005 12:00AM	1	96	40	11.0	-0.5	-0.5
10	Jan 4 2005 3:00AM	1	93	40	11.0	-0.69999999999999996	-0.69999999999999996
11	Jan 5 2005 12:00AM	9	97	20	6.0	2.2999999999999998	0.0
12	Jan 5 2005 3:00AM	0	95	30	8.0	.69999999999999996	.40000000000000002
13	Jan 6 2005 12:00AM	1	97	310	15.0	.5	.5
14	Jan 6 2005 3:00AM	2	97	290	11.0	-0.10000000000000001	-0.20000000000000001
15	Jan 7 2005 12:00AM	3	94	200	14.0	-0.5	-0.5
16	Jan 7 2005 3:00AM	4	97	230	13.0	-0.5	-0.5
17	Jan 8 2005 12:00AM	1	96	290	10.0	2.2000000000000002	2.2000000000000002
18	Jan 8 2005 3:00AM	2	97	250	11.0	1.1000000000000001	1.0
19	Jan 8 2005 9:00PM	9	97	330	12.0	4.2000000000000002	3.5
20	Jan 9 2005 12:00AM	1	96	350	12.0	2.5	2.5
21	Jan 9 2005 3:00AM	2	97	350	10.0	3.7000000000000002	3.3999999999999999
22	Jan 9 2005 9:00AM	7	97	340	10.0	3.6000000000000001	3.6000000000000001
23	Jan 9 2005 12:00PM	7	97	20	1.0	9.0999999999999996	6.2000000000000002
24	Jan 9 2005 3:00PM	9	97	350	11.0	7.5	3.7000000000000002
25	Jan 10 2005 12:00AM	1	96	360	4.0	3.3999999999999999	3.3999999999999999
26	Jan 10 2005 3:00AM	0	92	10	8.0	3.2999999999999998	3.2999999999999998
27	Jan 11 2005 12:00AM	2	97	260	10.0	3.2000000000000002	3.2000000000000002
28	Jan 11 2005 3:00AM	4	97	250	8.0	3.2000000000000002	3.2000000000000002
29	Jan 12 2005 12:00AM	5	97	250	13.0	4.0999999999999996	-0.20000000000000001

Abbildung 2.1: Beispieldaten

In Abbildung 2.1 sieht man das Ergebnis einer Abfrage auf eine meteorologische Datenbank des AWI. Schnell ist festzustellen, dass es sich in erster Linie um verschiedenste Zahlenformate handelt, von Fließkommazahlen (*Float*) über ganze Zahlen (*Integer*) bis hin zum Datum - in seltenen Fällen tauchen auch Buchstaben auf (Y(es) oder N(o) als Kennzeichnung für das Eintrittsverhalten eines bestimmten Ereignisses).

[6] vgl. [Wiki05 f]

Als Grundlage dieser Diplomarbeit dienen hier die Daten der 3-stündigen synoptischen Beobachtung[7], gemessen in den beiden Stationen „Neumayer[8]“ und „Koldewey[9]“ sowie auf den Reisen des Forschungsschiffes Polarstern.

Die Daten der „Ballonaufstiege“ (z.B. Ozonwerte), auch als „Upper Air Soundings“ bezeichnet, sowie die „Surface Radiations[10]“- und „Poldat[11]“-Daten werden aufgrund des prototypischen Charakters der Arbeit nicht behandelt. Sie werden im Nachhinein für das System aufbereitet und eingepflegt.

2.1.3 Publikationen

Mit dem Begriff Publikation ist entweder der Vorgang der Veröffentlichung eines Mediums oder das veröffentlichte Medium selbst gemeint. Publikationen können z.B. sein:

- Bücher
- Zeitschriften
- Tonträger
- Webseiten

Wissenschaftliche Publikationen sind schriftliche Arbeiten von mindestens einem Wissenschaftler. Diese Arbeiten unterliegen strengen formalen und inhaltlichen Kriterien, um von den sog. „Peers“ (engl. für „Ebenbürtige“) akzeptiert zu werden. Die Arbeiten werden dabei von Sachverständigen auf ihre wissenschaftliche Qualität überprüft.[12] Publiziert werden wissenschaftliche Veröffentlichungen meist als Bücher, Artikel in Fachzeitschriften oder in Konferenzbänden.

Publikationen spielen für Wissenschaftler eine herausragende Rolle. Meist werden sie von ihren Kollegen, der Fachwelt oder der Öffentlichkeit anhand der Qualität und Quantität (Stichwort: *publizierfreudig*) ihrer Veröffentlichungen gemessen. Davon hängt nicht nur die Finanzierung weiterer Studien oder Forschungsreihen ab, sondern auch ihr Ruf als wissenschaftliche Fachkraft.

[7] Messung von Temperatur, Luftdruck, Windrichtung, Windgeschwindigkeit etc.
[8] Ekström-Schelfeis, Atka-Bucht, nordöstliches Weddell-Meer – Position: 70°39'S, 08°15'W
[9] Spitzbergen, Königsfjord, Ny-Ålesund – Position: 78,9°N, 11,9°O
[10] Oberflächenstrahlung
[11] spezielle Messungen der Polarstern
[12] vgl. http://de.wikipedia.org/wiki/Peer-Review

2.1.4 Identifikation digitaler Objekte

Viele im Internet bzw. in Intranetzen abgelegte Objekte können mittels einer Adresse angesprochen / aufgerufen werden – eine Möglichkeit ist der Uniform Resource Identifier, kurz URI. Diese Variante setzt voraus, dass sich diese Adresse nicht ändert, was allerdings durch die Flüchtigkeit der Ablage von Inhalten auf Adressen im Internet sehr selten gegeben ist. Man benötigt daher ein System, welches nicht den Ort oder die Ressource des Objekts, sondern das Objekt selbst identifiziert - natürlich in Verbindung mit einer Adresse, denn neben einer eindeutigen Identifizierung muss das Objekt auch auffindbar sein.[13]

2.1.4.1 Digital Object Identifier (DOI)

Dieses System ist der Digital Object Identifier (engl. für „Bezeichner digitaler Objekte“), kurz DOI. Mittels DOI ist eine eindeutige und permanente Identifikation digitaler Objekte möglich. Das System von DOI ist vergleichbar mit ISBN/ISSN, geht jedoch aufgrund seiner integrierten Lokalisierungsfunktion darüber hinaus.

Für den Zugriff auf die identifizierten Objekte stellt die International DOI Foundation[14] (Betreiber des DOI-Systems), ein System zur Verfügung, mit welchem dem DOI der aktuelle Standort des Objekts zugeordnet wird. Ändert sich nun der Standort eines solchen Objekts, hat dies keinen Einfluss auf den DOI, es muss lediglich die Zuordnung in der DOI-Datenbank aktualisiert werden.[15]

Beispiel: **http://dx.doi.org/10.1594/PANGAEA.51609** führt zu der URI: http://doi.pangaea.de/10.1594/PANGAEA.51609

2.1.4.2 Persistent Identifier (PID)

DOI gehört zu den so genannten *Persistent Identifier*. Diese Bezeichner identifizieren *eindeutig* einen Eintrag in einem Repositorium.[16] Ziel dieser PID ist die Schaffung dauerhafter Adressierungsmechanismen. PID's in der Form „awi:Gro2000b“ werden innerhalb der AWI-internen Publikationsverwaltung ePIC (electronic Publication Information Center) verwendet. Diese sind allerdings nur lokal, sprich im Netz des Instituts eindeutig bzw. einzigartig, jedoch nicht weltweit. Daher können sie nur bedingt der Gruppe der Persistent Identifier (PID) zugeordnet werden.

[13] vgl. [Wiki05 d]
[14] vgl. http://www.doi.org
[15] vgl. [Wiki05 d]
[16] weitere Informationen auf http://www.openarchives.org/OAI/openarchivesprotocol.html unter Punkt 2.4

Der Schlüssel ist nach den URI-Richtlinien[17] (<*Schema*>:<*Schema-spezifischer Teil*>) aufgebaut. Am AWI besteht er aus einem Präfix (awi) nebst Trennsymbol (:), den ersten drei Buchstaben vom Nachnamen des Autors sowie einer vierstellige Ziffer (meist das Jahr der Veröffentlichung). Im Falle mehrerer Veröffentlichungen desselben Autors wird ein Buchstabe an den Schlüssel gehängt und ggf. inkrementiert.

Durch die Eindeutigkeit des PID ist es möglich, eine Publikation mit ihren Details aufzurufen. Das wird später in der Entwicklung des Portals von Nöten sein.

Beispiel: **awi:Gro2000b** → Publikation im Alfred-Wegener-Institut, Hannes Grobe im Jahr 2000, zweiter Eintrag

2.2 Datenhaltung

Die Daten, welche mittels des Prototypen gefunden werden können, sind an zwei Stellen aufbewahrt. Die Daten der Meteorologie sind in *Pangaea* archiviert, die Publikationen bzw. deren Metadaten, in *Fedora*. Diese beiden Systeme werden nun genauer erläutert.

2.2.1 Pangaea

Die in der umweltgeowissenschaftlichen Forschung anfallenden Daten machen ein umfassendes Datenmanagement erforderlich. Der Schwerpunkt liegt zur Zeit auf historischen, geologischen und marinen Daten. Es gilt, die Vielzahl an Datenarten und -formen in konsistente Formate zu überführen. Pangaea wurde geschaffen, um diese Arbeit zu unterstützen. Die Daten sollen langfristig archiviert und allgemein verfügbar gemacht werden. Vor allem dient es als wissenschaftliches Werkzeug zur Auswertung dieser Daten. Mit der Entwicklung von Pangaea wurde 1993 durch eine Gruppe von Wissenschaftlern und Informatikern am Alfred-Wegener-Institut in Zusammenarbeit mit anderen, an der Paläoklimaforschung beteiligten Instituten begonnen. Nachdem das System um die Erfassung beliebiger in Raum und Zeit geocodierbarer Daten erweitert wurde, erfolgte die Namensgebung in Anlehnung an die Systemstruktur nach dem Superkontinent, in dem alle Kontinente vor 200 Millionen Jahren vereint waren: Pangaea.[18]

Das Informationssystem basiert auf einer relationalen Datenbank und arbeitet nach dem Client/Server-Prinzip. Durch die Verbindung eines flexible Datenmodells mit einer Geocodierung der Daten lassen sich nahezu alle in der naturwissenschaftlichen Grundlagenforschung anfallenden geographisch und/oder zeitlich

[17] s. RFC 1630 unter http://www.ietf.org/rfc/rfc1630.txt
[18] vgl. [Pan05 a]

einzuordnenden Daten erfassen. Die Datenbankmanagementsoftware Sybase (Adaptive Server Enterprise 12.5) läuft auf einer SUN E10K mit 22 Prozessoren á 400 Mhz und 16 GB Hauptspeicher. Das zentrale Datenmodell ist mit zusätzlichen Tabellen ausgestattet. So enthalten z.B. die Tabellen *Staff* und *Institution* die Namen und Adressen der am System beteiligten Wissenschaftler. In *Reference* finden sich die Zitate zu Veröffentlichungen über Daten oder Expeditionen. *Parameter* hingegen beinhaltet die Namens- und Definitionsliste der speicherbaren analytischen Daten.

Der Import von Daten erfolgt über standardisierte Dateiformate (Tabellen, CSV usw.), welche von jedem Nutzer selbst angelegt werden können. Die Zuordnung der Metadaten zu den einzelnen Datensätzen erledigt eine Importroutine, die Zuordnung der Daten zu den Parametern erfolgt über eine Identifikationsnummer (ID). Der Import selbst ist (und bleibt) in der Regel immer einem Projekt-Datenkurator vorbehalten, um auf Dauer einen konsistenten Datenbestand zu gewährleisten. Zur Zeit befinden sich ca. 247.000 Publikationen in Pangaea.[19]

Das interne XML ist als sog. *binary large object* (kurz: blob[20]) in einer Tabelle gespeichert, welche mit den Datensätzen verbunden ist. Für die hohe Geschwindigkeit bei dem Zugriff auf die Metadaten sorgt Sybase EFTS, eine *full text search engine* (engl. für „Volltextsuchmaschine“). Diese XML-blobs können direkt über XSLT in andere Formate überführt werden (z.B. ISO 19115 Metadatenformat, Dublin Core oder die für die Pangaea-Ergebnisliste im Portal verwendeten Vorschaumetadaten).

2.2.2 Fedora

Das Akronym Fedora steht für *Flexible (and) Extensible Digital Object Repository Architecture*. Fedora ist eine Open-Source-Software, mit deren Hilfe man digitalen Inhalt in Archiven verwalten und zugänglich machen kann. Das Repositorium vereinigt Elemente von Content Management, Electronic Publishing, Digital Library Management, Records Management, Versionsmanagement, Digital Asset Management, Dokumentenmanagement und Archivierung in einem System.[21]

Der Kern von Fedora ist ein konsequent durchgesetzter Objekt-Ansatz:

- Persistent Identifier (PID) für die (Informations-)Objekte
- Steuerung der Nutzung erfolgt durch Metadaten

[19] vgl. [Pan05 a]

[20] Ein *Binary Large Object* ist ein binärer Datentyp, welcher besonders für die Speicherung großer Datenmengen geeignet ist. Gekennzeichnet ist der *Blob* in einer Datenbank durch ein Datenfeld.

[21] vgl. [CM05]

- systeminterne Metadaten
- beliebige Inhaltskomponenten
- Inhaltsverwaltung durch Kombination von XML-Repositories und relationale Datenbanken

Zu den Eigenschaften gehören u.a. der durchgängige Einsatz von XML, die Unterstützung des OAI-PMH[22] und Standards wie Dublin Core, eine flexible Skalierbarkeit aufgrund der Verteilbarkeit der Objekte sowie eine breite API[23]. Nicht zu vergessen sind die SOAP[24]-basierenden Serviceschnittstellen, welche die Web Service-Fähigkeit von Fedora ermöglichen. Hauptzielgruppe sind derzeit Bibliotheken, jedoch wird der Einsatz für Verlage und Archive immer interessanter. Die häufigste Anwendung findet Fedora in den Universitäten der USA, das internationale Interesse wächst aber stetig an.[25]

Die Entwicklung begann 1997 an der Cornell-Universität und wurde von mehreren Organisationen gefördert. 1999 kam die erste Version in der Digital Library der Universität von Virginia zum Einsatz. Als Open Source wurde Fedora 2002 freigegeben.

Die Implementierung von Fedora am Alfred-Wegener-Institut erfolgt im Rahmen einer parallel zu dieser laufenden Diplomarbeit. Gespeichert und verwaltet werden hierbei verschiedene Publikationsdaten sowie die persönlichen Internetseiten der AWI-Mitarbeiter[26]. Zur Zeit befinden sich ca. 14.000 Publikationen in Fedora.

2.3 Web Services

Der wichtigste Punkt und technische Basis dieser Diplomarbeit sind Web Services. Wohl kaum eine andere Technologie hat in den letzten Jahren für mehr Aufsehen gesorgt als diese. Viele Unternehmen sind dabei, die noch „junge“ Architektur in ihre Systeme zu implementieren, oder gar ihre Infrastruktur und Prozesse an Web Services anzupassen. Was sich hinter Web Services verbirgt, darüber geben die nachfolgenden Punkte Aufschluss.

[22] Open Archives Initiative Protocol for Metadata Harvesting
[23] Application Programming Interface
[24] s. Untersektion 2.3.4 auf Seite 20
[25] vgl. [Fed05]
[26] z.B. http://www.awi-bremerhaven.de/People/show?bbraeuer

2.3.1 Extensible Markup Language (XML)

Die Extensible Markup Language ist ein Standard zur Erstellung (baumartig) strukturierter, maschinen- und menschenlesbarer Dokumente. XML definiert hierbei die Regeln für den Aufbau solcher Dokumente. Daten, Struktur und Format von Dokumenten werden separat behandelt. Das Regelwerk wird dabei in der Dokument Type Definition (DTD) festgelegt. Meteorologische Daten können so z.B. als Tabelle und als Grafik ausgegeben werden, weil beide Anwendungen dieselbe XML-Datenbasis verwenden. Weiterhin kann XML (mit Extensible Stylesheet Language Transformation (XSLT)) in andere Dokumentenformate umgewandelt werden (z.B. HTML, RTF, PDF, SVG, (formatierter) Text usw.).

Bei einer XML-Anwendung müssen die Details der jeweiligen Dokumente spezifiziert werden. Insbesondere betroffen ist davon die Festlegung der Strukturelemente und ihre Anordnung innerhalb eines Dokumentenbaums. Mittels des Standards XML lassen sich beliebige, in ihrer Grundstruktur jedoch stark verwandte Auszeichnungssprachen definieren. Die Extensible Markup Language ist eine vereinfachte Teilmenge der Standard Generalized Markup Language (SGML).[27]

> „Die Extensible Markup Language (XML) ist eine Teilmenge von SGML, die vollständig in diesem Dokument beschrieben ist. Das Ziel ist es, zu ermöglichen, generic SGML in der Weise über das Web auszuliefern, zu empfangen und zu verarbeiten, wie es jetzt mit HTML möglich ist. XML wurde entworfen, um eine einfache Implementierung und Zusammenarbeit sowohl mit SGML als auch mit HTML zu gewährleisten.“[28]

```
1 <?xml version="1.0"?>
2 <wurzelelement>
3    <unterelement>Hello World!</unterelement>
4 </wurzelelement>
```

Quelltext 2.1: Beispiel: XML-Dokument

2.3.2 Allgemeine Beschreibung von Web Services

> „A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols.“[29]

[27] vgl. [Wiki05 j]
[28] s. [W3C04]
[29] s. [W3C02]

Aus der Definition des W3C kann man herauslesen, dass es sich bei einem Web Service um eine Software-Anwendung handelt, welche mit einem Uniform Resource Identifier (URI) identifizierbar ist. Deren Schnittstellen sind als XML-Artefakte definiert, beschrieben und können gefunden werden. Ein Web Service unterstützt die direkte Interaktion mit anderen Software-Agenten. Dabei finden XML-basierte Nachrichten Verwendung, welche über internetbasierte Protokolle ausgetauscht werden. Web Services spielen im Bereich E-Business als Middleware eine zunehmend bedeutende Rolle. In Abbildung 2.2 sieht man den Aufbau eines Web Services, beginnend vom eigentlichen Dienst (Service) bis hin zu seiner Veröffentlichung in einem Verzeichnisdienst.

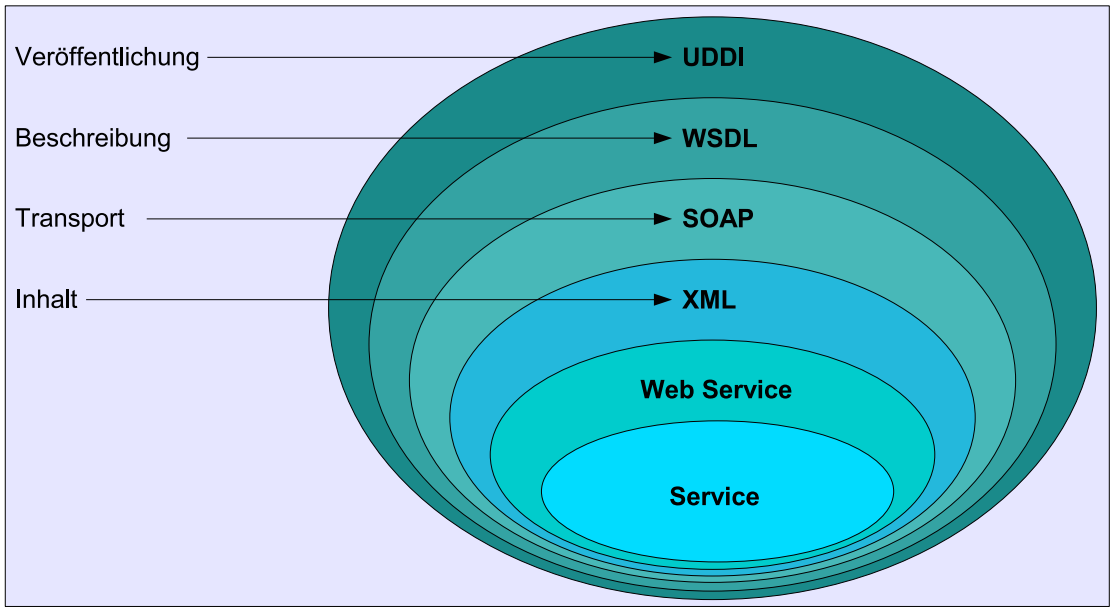


Abbildung 2.2: Zwiebelschalenmodell Web Service

Spricht man aber von den Web Services, welche seit einiger Zeit in Fachzeitschriften oder auf IT-Konferenzen als Ablöser für die Client/Server-Architektur im Gespräch sind, muss die Definition erweitert werden. Die Web Services sind also Applikationen, welche mit einem Client über das XML-basierende Protokoll SOAP kommunizieren und dabei ein beliebiges Trägerprotokoll (HTTP, FTP ...) zum Transport der Informationen verwenden.[30]

Web Services sorgen für Interoperabilität zwischen verschiedenen Applikationen, welche auf ungleichen Plattformen aufgesetzt sind. Die verwendeten, offenen und standardisierten Protokolle sind (wenn möglich) text-basierend gehalten, um den Entwicklern das Verständnis zu erleichtern. Durch die Benutzung von HTTP können Web Services durch Firewalls angesprochen werden, ohne die Regeln der Firewalls zu ändern. Anwendungen und Dienste von verschiedenen Unternehmen

[30] vgl. [Langner 2003, S. 73]

und Standorten können leicht kombiniert werden. Web Services erlauben die Wiederverwendung von Diensten und Komponenten innerhalb einer Infrastruktur. Abbildung 2.3 zeigt das Zusammenspiel der drei Web-Service-Parteien „Service Requester“ (Client) , „Service Provider“ (Service) und „Service Broker“ (Verzeichnisdienst) mit den Standards WSDL, UDDI und SOAP.

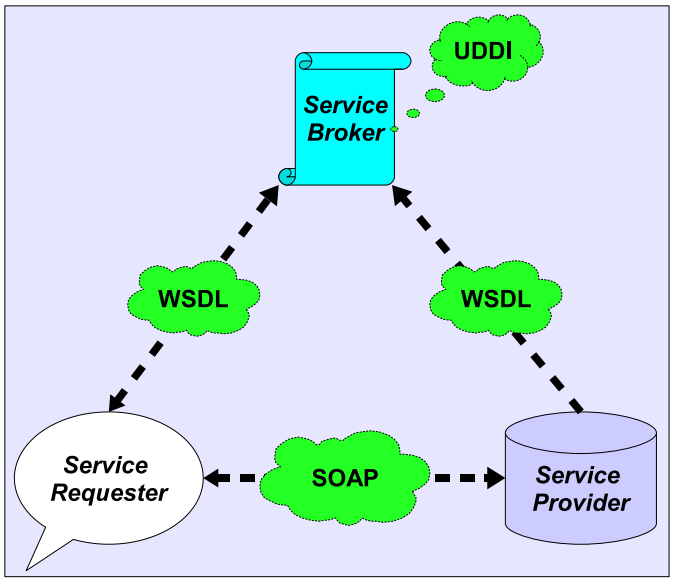


Abbildung 2.3: Zusammenspiel der Web-Service-Komponenten, vgl. [Wiki05 i]

Es gibt viele gute Gründe, welche für den Einsatz von Web Services sprechen. Zu den Wichtigsten zählen:

- Verwendung von HTTP über TCP Port 80
  - Im Gegensatz zu vergleichbaren Technologien wie CORBA, DCOM oder auch Java RMI treten beim Einsatz von Web Service keine Probleme mit Firewalls und deren Konfigurationen auf, da der Port 80 (der Standardport für Web Browser) stets in irgendeiner Art und Weise offen ist. Web Services „tunneln“ alles durch den Port 80.

    Allerdings sind sie nicht an HTTP gebunden, sondern arbeiten auch mit anderen Protokollen wie FTP oder SMTP zusammen. Dies kommt wiederum der Erweiterung ihrer Einsatzgebiete zu Gute.

- Verwendung von Standards
  - Web Services bieten durch den Einsatz verbreiteter und bereits bestehender Standards eine offene und flexible Architektur. Sie ist unabhängig von den verwendeten Plattformen, Programmiersprachen, Protokollen et cetera.

Ein Windows-C#-Client, der hinter eine Firewall steht, kann mit einem Java-Server, welcher auf einem Sun Solaris-System implementiert ist, kommunizieren. Die Standards ermöglichen eine hohe Interoperabilität über jedwede Heterogenität im Internet hinweg.

- Keine Lizenzkosten
  - Durch die offenen Standards entfallen Lizenzkosten. Da diese Standards meist weit verbreitet sind, können sie auch in vielen Bereichen eingesetzt werden, was wiederum die Kosten senkt.

Dennoch existieren auch (noch) einige Nachteile:

- mangelnde Sicherheit
  - Grundsätzlich liegen die Daten beim Transport über SOAP im Klartext vor, d.h. sie sind theoretisch von jedem einsehbar. Darum ist der Einsatz von Verschlüsselung- und / oder Authentifizierungstechniken erforderlich. Beispiele dafür wären WS-Security oder SAML.
- unter Umständen negative Leistung
  - Die Performanz eines Web Services kann durch das „Parsen[31]“ großer XML-Dateien negativ beeinflusst werden. Bei verteilten Systemen kann der Verwaltungsaufwand erheblich zunehmen.

2.3.3 Web Service Description Language (WSDL)

Durch den plattform-, programmiersprachen- und protokollunabhängigen XML-Standard WSDL lassen sich Web Services beschreiben. Mit Hilfe von WSDL kann festgelegt werden, welche Methoden der Client aufrufen kann bzw. der Service anbietet, welche Parameter mit übergeben werden müssen und was für Rückgabewerte die Methoden zurückliefern.

Eine Web Service-fähige Sprache verarbeitet die WSDL-Beschreibung und bietet dem Entwickler so die Möglichkeit, auf die angebotenen Methoden zuzugreifen. Die Beschreibung besteht dabei aus fünf XML-Hauptelementen (s. auch Abb.2.4):

- types – definiert die Datentypen, welche zum Austausch der *messages* benutzt werden
- messages – beschreibt eine Nachricht, welche vom Client zum Server geschickt wird und anschließend vom Server zum Client gelangt

[31] Analysieren der übergebenen Zeichenstruktur und Übersetzen in eine neue Struktur (z.B. XML-Baumstruktur)

- portType – Kommunikationsform, welche der/den *messages* zugeordnet wird
  - one-way: Server empfängt Nachricht von Client
  - request-response: Server empfängt Nachricht, sendet dann Nachricht an Client
  - solicit-response: Server sendet Nachricht, empfängt dann Nachricht an Client
  - notification: Server sendet Nachricht an Client
- binding – Bindung an ein konkretes Protokoll und Datenformat
- service – fasst alle *port*-Elemente, welche für jedes *binding*-Element genau eine URI spezifizieren, zu einer Gruppe zusammen

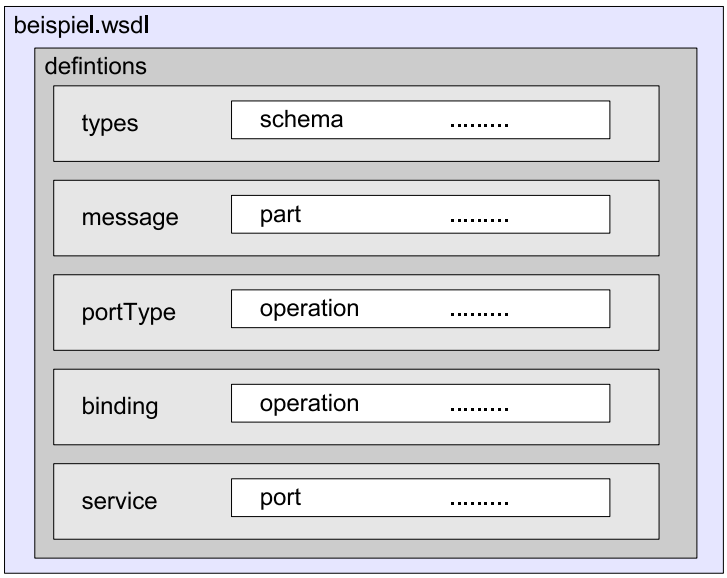


Abbildung 2.4: Aufbau einer WSDL-Beschreibung

2.3.4 Simple Object Access Protocol (SOAP)

SOAP ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls (RPC) durchgeführt werden können. Es legt die Formatierung fest, mit der die Information, welche von einem Rechner zum anderen (z.B. Client zu Service Provider oder umgekehrt) übertragen werden, kodiert werden müssen, um eine Verständigung zwischen den Rechnern zu ermöglichen. Dabei stützt sich SOAP auf XML, so dass es von Standard-Parsern gelesen werden kann.

Das Protokoll kümmert sich wie erwähnt nur um die Formatierung der Daten, nicht wie sie von einer Stelle zur anderen gelangen oder RPC's / Services auf dem

Service Provider ausführen. Das ist eine Stärke der Web Services: Wie ein Aufruf ausgeführt, darum muss sich der Provider kümmern. Der Nutzer des Web Service braucht lediglich zu wissen, wo er den Dienst wie aufruft.

SOAP ist ähnlich einem Brief aufgebaut (s. Abb. 2.5). Ein Briefumschlag („Envelope“) umfasst die Nachricht. Diese besteht, eingebettet in den *Envelope*, aus zwei Teilen. Der Kopfzeile („Header“) enthält z.B. Absenderinformationen. Die eigentliche Nachricht, welche übermittelt werden soll, ist im „Body“ untergebracht. Eine vollständige SOAP-Nachricht zeigt der Quelltext 2.2.

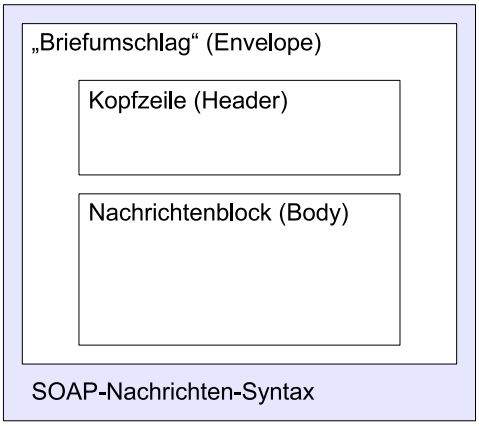


Abbildung 2.5: Aufbau einer SOAP-Nachricht, vgl. [Vbip01]

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               soap:encodingStyle="http://schemas.xmlsoap.org/soap/
                                   encoding/">
  <soap:Header>
     <h:from xmlns:h="http://www.wrox.com/Header">SoapGuy@wrox.com</h:from>
  </soap:Header>
  <soap:Body>
     <w:GetSecretIdentity xmlns:w="http://www.wrox.com/heroes/">
        <w:codename>XSLT-Man</w:codename>
     </w:GetSecretIdentity>
  </soap:Body>
</soap:Envelope>
```

Quelltext 2.2: Beispiel: SOAP-Nachricht

2.4 Entwicklung

Nach Abhandlung der Datensektion widmet sich dieser Teil der Entwicklung. Neben der Beschreibung der Entwicklungsumgebungen und verwendeten Werkzeuge, kommen auch Erläuterungen der Programmiersprachen, welche für die Programmierung unerlässlich sind sowie Mittel zur Gestaltung des Portals zur Sprache.

2.4.1 Entwicklungsumgebung

Die Entwicklung der Anwendung erfolgt auf zwei verschiedenen Umgebungen. Das Testsystem verfügt über einen 1,8 GHz Pentium IV Prozessor, 1024 MB Arbeitsspeicher sowie das Betriebssystem Windows 2000 Professional der Firma Microsoft.

Als lokaler Server für die Anwendung läuft der beliebte Open Source Web Server Apache in der Version 2.0.53 in Verbindung mit PHP 5.0.4 (siehe Untersektion 2.4.3 auf Seite 24). Um eine einfache Installation der Komponenten zu gewährleisten, wurde das Paket XAMPP 1.4.13 des Apache Friends-Projekts[32] zur Förderung und Verbreitung des Apache Web-Servers und damit verbundener Technologien wie MySQL, PHP und Perl eingesetzt. Dieses beinhaltet die eben genannten Anwendungen vorkompiliert und -konfiguriert.

Allerdings kommt man bei einer komplexen Anwendung wie der zu dokumentierenden Entwicklung nicht daran vorbei, die Applikationen selbst zu konfigurieren, um die gewünschten Resultate zu erreichen. Dazu ist anzumerken, dass viele der mitgelieferten Einstellungen Standardeinstellungen sind, welche auch auf zahlreichen Produktivsystemen weltweit zum Einsatz kommen (Apache ist mit fast 70% Marktführer im Bereich WebServer[33]).

Das zweite Test- und spätere Produktivsystem ist ein Sun Fire 15K Server[34], ausgestattet mit 64 UltraSPARC III Cu Prozessoren á 900-MHz. Genutzt werden unter der Testdomain „http://web.awi-bremerhaven.de/php/“ allerdings nur vier davon – was für Testzwecke aber mehr als ausreichend ist. Bei Bedarf können weitere Prozessoren zugeschaltet werden. Die SF15K ist mit 128 GB Arbeitsspeicher gut ausgestattet. Benutzt werden für diese Zwecke aber wieder nur 8 GB. Als Betriebssystem kommt Sun Solaris in der Version 9 zum Einsatz.

Installiert sind der Apache (Ver. 2.0.54) und PHP (Ver. 5.0.4), wie auf dem Testsystem in der Version. Die Änderungen zwischen den Apache-Versionen 2.0.53 und 2.0.54 sind für die Entwicklung irrelevant.

[32] vgl. http://www.apachefriends.org/
[33] vgl. [Ncr05]
[34] vgl. http://www.sun.com/servers/highend/sunfire15k/

Die eigentliche Programmierung erfolgt mittels *Notepad++*[35] *(Ver. 3.1)*, einem Open Source-Texteditor mit Hervorhebungsfunktion für zahlreiche Programmiersprachen. Zur Überprüfung der Funktionalität und der Optik der Ausgabe werden folgende Browser verwendet:

- Microsoft Internet Explorer 6 SP1 (de)
- Mozilla FireFox 1.0.6 (de) & Deer Park Alpha 2 (en)
- Opera 8.01 (de)
- Safari 1.3 & 2.0 (de)

Diese Browser werden bei über ca. 95% der Internetbenutzer verwendet. Viele andere Browser verhalten sich konform zu einem von den genannten, was mögliche Inkompatibilitäten weitestgehend ausschließt.

Für die Erstellung des schriftlichen Teils der Diplomarbeit kam LaTeX 2_ε (Windows Version von TeX: MiKTeX[36] 2.4) sowie das TeXnicCenter[37] (Ver. 1 Beta 6.30) zum Einsatz. Die Bilder dieses Dokumentes wurden bearbeitet / konvertiert mittels *GNU Image Manipulation Program*[38], kurz *The GIMP* (Version 2.2).

2.4.2 Perl

Larry Wall entwarf 1987 mit Perl eine freie, plattformunabhängige Programmiersprache, welche auch als Skriptsprache oder *dynamic language* bezeichnet werden kann. Der Linguist entwickelte sie als Synthese aus C, den UNIX-Befehlen und anderen Einflüssen. Der Einsatz war ursprünglich als Werkzeug zur System- und Netzwerkadministration gedacht, mittlerweile hat Perl bei der Entwicklung von Webanwendungen und in der Bioinformatik weite Verbreitung gefunden.[39]

Die Sprache bietet für viele Probleme eine schnelle Lösung und eine große Freiheit für Programmierer. Zu den Stärken der Sprache gehören u.a. der Umgang mit Texten bzw. Textdateien sowie die vielen frei verfügbaren Module. Im Laufe der Zeit haben sich zwei Backronyme für Perl durchgesetzt: *Practical Extraction and Report Language* und *Pathologically Eclectic Rubbish Lister*.[40]

[35] vgl. http://notepad-plus.sourceforge.net/
[36] vgl. http://www.miktex.org
[37] vgl. http://www.toolscenter.org/
[38] vgl. http://www.gimp.org/
[39] vgl. [Wiki05 g]
[40] vgl. [Wiki05 g]

Für die Entwicklung wird die Version 5 der Sprache verwendet. Als eine der besagten Erweiterungen kommt *MIME::Lite*[41] zum Einsatz. Dieses umfangreiche Modul ermöglicht das komfortable Versenden von Emails mit Anhängen mittels Skript.

Die Tatsache, dass Perl sehr gut mit UNIX-Systemen harmoniert, hat zu dem Entschluss geführt, es für diese Diplomarbeit zu verwenden. Perl selbst ist leicht lern- und wartbar, die Notwendigkeit dieser Charakteristika wird im späteren Verlauf der Arbeit noch erläutert.

```
1 #!/usr/local/bin/perl
2 print "Hello world!";
```

Quelltext 2.3: Beispiel: Perl-Skript

2.4.3 PHP Hypertext Preprocessor (PHP)

PHP, das rekursive Akronym für „PHP: Hypertext Preprocessor“ (ursprünglich „Personal Home Page Tools“) wurde 1995 als Sammlung verschiedener Perlskripte entwickelt, daher auch die teilweise starke Ähnlichkeit mit Perl. Was folgte war die Umwandlung der Skriptsammlung mittels C durch Rasmus Lerdorf, worin PHP auch heute noch geschrieben ist. Die Sprache gehört zu den Skriptsprachen, mit denen sich dynamische Webseiten realisieren lassen. Bei PHP handelt es sich um Open-Source-Software.[42]

Noch mehr als Perl zeichnet sich PHP besonders durch die leichte Erlernbarkeit, die breite Datenbankunterstützung und Internet-Protokolleinbindung sowie die Verfügbarkeit zahlreicher zusätzlicher Funktionsbibliotheken aus. Inzwischen ist PHP bei der Version 5 angekommen, was unter anderem auch ernsthaftes objektorientiertes Programmieren durch das Hinzufügen vieler Sprachkonstrukte ermöglicht.[43]

Die Funktionsweise von PHP ist in Abbildung 2.6 erläutert. In einem Client (z.B. Browser) wird eine Anfrage für eine PHP-Datei gesendet, z.B. durch Aufruf einer URL, Anklicken eines Links oder Buttons usw. Die Anfrage wird an den entsprechenden Server weitergeleitet (1). Der Server empfängt die Anfrage und lädt (sofern vorhanden) die geforderte PHP-Datei (2). Diese wird an den PHP-Interpreter übergeben (3). Der Interpreter durchläuft die Datei (4) und liefert, je nach Erfolg oder Misserfolg der Ausführung das Ergebnis (bei Browsern meist eine HTML-Datei, aber auch Bilder, XML, PDF etc. sind möglich) zurück (5). Der WebServer gibt nun dieses Ergebnis wieder an den Client (6) und der Vorgang ist abgeschlossen.

[41] vgl. http://www.zeegee.com/code/perl/MIME-Lite/
[42] vgl. [Wiki05 h]
[43] vgl. [Wiki05 h]

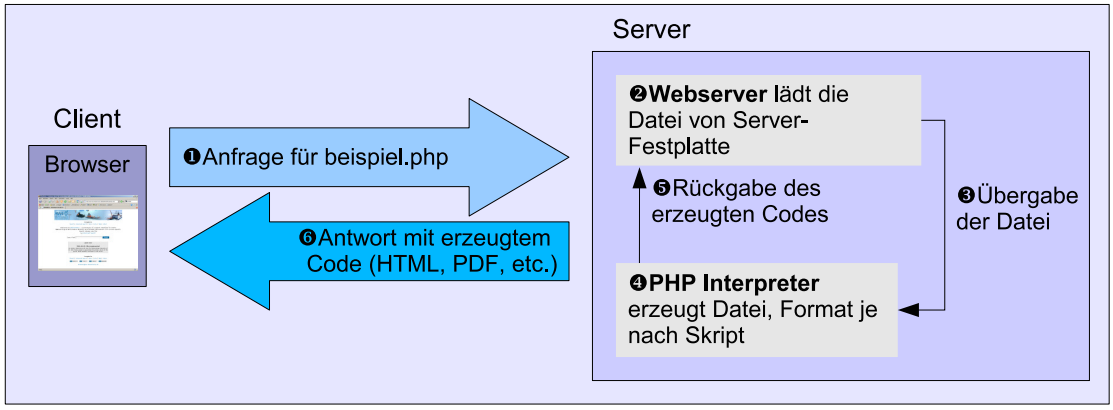


Abbildung 2.6: Funktionsweise PHP, vgl. [Wiki05 i]

```
1 <?php
2     echo "Hello world!";
3 ?>
```

Quelltext 2.4: Beispiel: PHP-Skript

2.4.3.1 Erweiterung: SOAP

Früher waren Web Service-Entwickler und -Anwender auf externe Module angewiesen, um die Web Service-Technologie unter PHP zu realisieren. Seit der Version 5 gehört zum PHP-Kern eine eigene Implementierung zur SOAP-Unterstützung. Es unterstützt die SOAP 1.1, SOAP 1.2 und WSDL 1.2 Spezifikationen. Diese Erweiterung bildet die Grundlage für das Verwenden von Web Services in dieser Diplomarbeit.

Mit PHP und der WSDL lassen sich schnell Ergebnisse realisieren. Ein einfaches Beispiel, welches den Dienst „BabelFish“ (ein Übersetzungswerkzeug) verwendet, soll dies demonstrieren:

```
1 <?php
2   $client = new SoapClient('http://www.xmethods.net/sd/2001/BabelFishService.wsdl
        ');
3   $result = $client->BabelFish('de_en', 'Hallo Welt');
4   echo "<h1>".$result."</h1>";
5 ?>
```

Quelltext 2.5: Beispiel: Web Service-Aufruf mit PHP

Nur zwei Zeilen bedarf es, um einen Web Service anzusprechen und zu nutzen. Der Klasse *SoapClient* wird die Adresse der WSDL-Beschreibung übergeben. Daraus wird ein Client generiert. Mit diesem Client lassen sich nun die im Web Service bereitgestellten Methoden ansprechen.

In dem Beispiel werden zwei Parameter übergeben. Zum Einen in welche Sprache übersetzt werden soll (hier: deutsch→englisch), zum Anderen der zu übersetzende

Text. Wird nun die Anfrage aus Quelltext 2.5 ausgeführt, erhält man das in Abb. 2.7 dargestellte, korrekte Ergebnis.

Abbildung 2.7: Web Service-Aufruf mit PHP

2.4.3.2 Erweiterung: SimpleXML

Ansätze für die Bearbeitung von XML mit PHP gab es schon früher. Viele dieser Erweiterungen sind jedoch nicht über das Beta-Stadium hinaus gekommen. Nun gibt es in PHP 5 zwei brauchbare Möglichkeiten zur Handhabung von XML.

Zum Einen wäre da DOM (Document Object Model), eines der mächtigsten Werkzeuge in PHP. Es erlaubt die Arbeit an einem XML-Dokument mittels der DOM API. Die Einarbeitungszeit und Nutzung dieser Erweiterung birgt jedoch deutlich mehr Aufwand und bringt nur Vorteile in „extremen“ Anwendungssituationen wie z.B. der Erstellung und Validierung „riesiger“ oder komplexer XML-Dokumente.

Die andere Erweiterung, kurz SimpleXML genannt, bietet einfache und leicht nutzbare Werkzeuge, um ein XML-Dokument in ein Objekt zu verwandeln. Diese Objekt lässt sich dann wie ein Array behandeln. SimpleXML hat nicht den Funktionsumfang von DOM, dennoch erfüllt es die Kriterien, die im Bezug auf XML an dieses Projekt gestellt werden. Das wären neben der Verarbeitung der von den Web Services gelieferten Ergebnisse auch Verständlichkeit und Wartbarkeit, damit nachfolgende Entwickler den Quelltext einfach handhaben können.

2.4.4 Gestaltung

Da es sich bei der Arbeit um ein öffentlich zugängliches (also über Internet verfügbares) Projekt handelt, kommen neben der Entwicklung der Kernkomponenten auch gestalterische Elemente zum Tragen. An dieser Stelle werden nur die Grundlagen der Programmierung des Portals behandelt. Erläuterungen zu den Aspekten

der Software-Ergonomie etc. finden in einem späteren Teil der Arbeit statt (s. Seite 60).

2.4.4.1 Portal

Bei einem Portal handelt es sich um ein Anwendungssystem, welches durch folgende Eigenschaften gekennzeichnet ist:

- Integration von Anwendungen, Prozessen und Diensten
- Bereitstellung von Funktionen zur Personalisierung, Sicherheit, Suche und Präsentation von Informationen

Diese Eigenschaften werden je nach Zweck eines Portals bei der Entwicklung integriert. So benötigt z.B. das zu entwickelnde Portal keine Funktionen für Personalisierung und Sicherheit, sondern es beschränkt sich auf die Bereitstellung der Suche (nach meteorologischen Daten) und die Präsentation der gefundenen Ergebnisse.

Es existieren zwei Arten von Portalen: vertikale und horizontale. Das Portal der Suchmaschine „Google“[44] (s. Abb. 2.8), welches hauptsächlich „nur“ die Dienstleistung der Suche anbietet (die anderen Funktionen seien hier ausgelassen), gehört zur Gruppe der vertikalen Portale. Diese bieten in erster Linie Informationen und Dienstleistungen zu einem speziellen Bereich an – in diesem Fall die Suche nach Begriffen im World Wide Web. Da es bei der Umsetzung des zu entwickelnden Portals ebenfalls um eine spezialisierte Funktionsleistung handelt, gehört es auch in die Kategorie der vertikalen Portale.

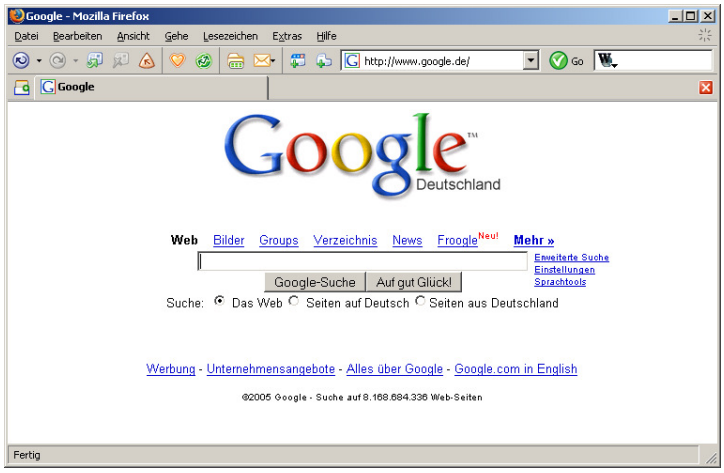


Abbildung 2.8: Vertikales Portal: Google

[44] vgl. http://www.google.de

Das zweite Bild (s. Abb. 2.9) zeigt das Portal des Webhosters und Internet Service Provider „Lycos“[45]. Es gehört zur Gruppe der horizontalen Portal. Diese sind gekennzeichnet durch ein breites Angebot (Nachrichten, Email, Chat, Internetrecherche etc.), meist in Verbindung mit einer Personalisierung der Benutzer. Durch die Möglichkeit, die Inhalte anzupassen, gehen horizontale Portal auf die individuellen Wünsche ihrer Anwender ein.

Abbildung 2.9: Horizontales Portal: Lycos

Bei Standard-Portalen kommen oft Anwendungen in Unterfenstern (sog. „Portlets“) zum Einsatz. Diese Portlets können vom Benutzer personalisiert werden, d.h. Inhalte entsprechend seinen Vorgaben und Wünschen enthalten. Meist können sie auch konfiguriert, minimiert oder entfernt werden. Des Weiteren wurden für Portlets zwei wichtige Spezifikationen definiert:

- Web Services for Remote Portlets (OASIS-Standard WSRP).
- Java Specification Request (JSR) 168

Zu WSRP ist zu sagen, das eine Verwendung von Web Services in Portlets möglich ist. Die Implementierung erfolgt ähnlich der in einer Website, nur das gewisse Eigenschaften der Portlets berücksichtigt werden müssen.

[45] vgl. http://www.lycos.de

Bei der Entwicklung eines Portals gibt es je nach Anforderungen zwei Möglichkeiten. Das Portal kann selbst entwickelt werden oder es kommt eine Portal-Standard-Software zum Einsatz. Ein selbst entwickeltes Portal verfügt über die größtmögliche Freiheiten ohne Vorgaben bei der Umsetzung. Von daher wurde sich bei der Lösung der Aufgabe für diese Möglichkeit entschieden. Der Einsatz eines „vorgefertigten“ Portals ist bei größeren Entwicklungen bzw. entsprechenden Einsatzgebieten gerechtfertigt.

2.4.4.2 Extensible Hypertext Markup Language (XHTML)

Die Extensible Hypertext Markup Language (XHTML) ist die Ablösung von HTML als W3C-Standard der Textauszeichnungssprache für Webseiten im World Wide Web. XHTML verwendet die strengere und einfacher zu parsende SGML-Teilmenge XML als Sprachgrundlage, womit XHTML-Dokumente den Syntaxregeln von XML entsprechen. Der Standard XHTML 1.0 enthält alle Elemente von HTML 4.01, was eine Umformung von HTML-4.01-konformen Seite möglich macht.

Da die HTML-Parser der verbreiteten Browser tolerant gegenüber Syntaxfehlern sind, kann auch ein nicht XHTML-fähiger Browser XHTML-Dokumente richtig darstellen, indem er sie als normales HTML verarbeitet. Da viele HTML-Dokumente im World Wide Web nicht dem formalen Standard entsprachen und gleichzeitig Fehlermeldungen von der Inakzeptanz der Benutzer betroffen waren, ist die besagte Fehlertoleranz das Ergebnis darauf. Durch die Verwendung von XML und dessen Grundideen eines unkomplizierten Datenaustausch und der problemlosen automatisierten Verarbeitung, sind Programme, welche XHTML verarbeiten immer weniger tolerant gegenüber einer nicht-konformen Struktur.[46]

Die aktuellste Version ist XHTML 1.1, die Version 2.0 befindet sich in der Entwicklungsphase und ist inzwischen als siebtes „Working Draft“ (engl. für „Arbeitspapier“) veröffentlicht worden. Diese Version beinhaltet u.a. die Spezifikation für eine weitere XML-Anwendung, welche die nächste Generation bzw. den zukünftigen Standard für Formulare im Netz beinhaltet - XForms[47]. Da im Portal mehrere Formulare zur Anwendung kommen, wurde die Verwendung dieses Standard mit in Erwägung gezogen. Allerdings wird XForms von den derzeitigen „Rendering Engines“ in den Browsern nicht unterstützt bzw. ist noch nicht ausgereift, weshalb es nur für zukünftige Entwicklungen und Erweiterungen in Frage kommt.

```
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" lang="de" xml:lang="de">
4   <head>
```

[46] vgl. [Wiki05 e]
[47] vgl. http://www.w3.org/TR/xforms/

```
5     <title>Hello</title>
6   </head>
7   <body>
8     <p>Hello World!</p>
9   </body>
10 </html>
```

Quelltext 2.6: Beispiel: XHTML-Dokument

2.4.4.3 Cascading Style Sheets (CSS)

Cascading Style Sheets ist eine Sprache zum Formatieren von z.B. XML-, HTML- oder XHTML-Elementen und eine unmittelbare Ergänzung zu HTML. Durch die Trennung von Stil und Inhalt wird das Veröffentlichen und Betreuen von Dokumenten wesentlich vereinfacht. CSS ermöglicht es auch, Inhalte dem jeweiligen Ausgabemedium (z. B. Druck, Projektion, Sprachausgabe etc.) entsprechend abzuändern. Oder um für ein Anzeigemedium wie einen PDA oder ein Mobiltelefon mit geringerer Auflösung die Anzeige zu optimieren (geringere Seitenbreite und -höhe). Verglichen mit den HTML-Formatierungen bietet CSS erheblich mehr Möglichkeiten, wie z.B. Schriftgestaltung, Rahmen, Innen- und Außenabstände, Listen, Hintergründe, Positionieren etc.[48]

CSS wird ebenfalls vom W3C spezifiziert, die CSS Level 1-Spezifikation von 1996 ist in aktuellen Browsern nahezu vollständig umgesetzt. CSS Level 2-Elemente sind auch bereits weit verbreitet, werden allerdings noch häufig fehlerhaft bzw. unterschiedlich in den Browsern umgesetzt und dargestellt. CSS Level 3 befindet sich derzeit in der Entwicklung. Cascading Style Sheets gilt heutzutage als die Standard-Stylesheet-Sprache für das World Wide Web.

```
1 html, body {
2   font-family: Verdana,Arial,Geneva,Helvetica,sans-serif;
3   font-size: 12px;
4   background: #FFFFFF;
5 }
```

Quelltext 2.7: Beispiel: CSS-Auszug

In dem Beispiel 2.7 wurde folgendes festgelegt:

- Schriftfamilie Verdana, falls diese nicht vorhanden, nutze Arial usw.
- Standardschriftgröße 12 Pixel
- Hintergrundfarbe Weiß

[48] vgl. [SH05; Wiki05 a]

3 Vorausgehende Analyse

3.1 Anforderungen an das System

Nach den Erklärungen der grundlegenden Systeme, Architekturen etc. folgt nun der erste Schritt in Richtung Entwicklung. Zu Beginn dieser steht eine Analyse, um herauszufinden, über welche Funktionalitäten das Portal verfügen muss, was nicht gebraucht wird, wie das Vorgehen auszusehen hat et cetera.

3.1.1 Entwicklungsmodell

Als Vorlage soll das in der professionellen Softwareentwicklung als Standard Verwendung findende V-Modell (s. Abb. 3.1) dienen.

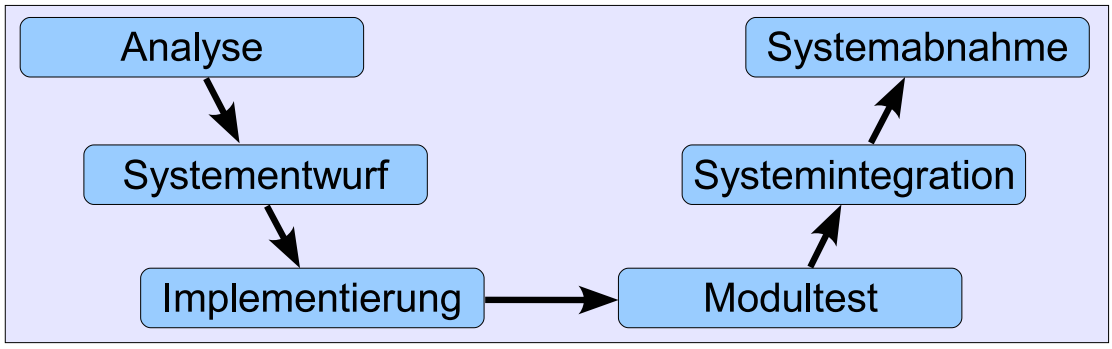


Abbildung 3.1: V-Modell, vgl. [Stein 2004, S. 41]

Das V-Modell ist das Vorgehensmodell zur Softwareentwicklung, welches den Entwicklungsstandard für IT-Systeme der öffentlichen Hand in Deutschland darstellt. Es werden im Gegensatz zu den klassischen Phasenmodellen lediglich Aktivitäten und Ergebnisse definiert, aber keine strikte zeitliche Abfolge gefordert. Es fehlen weiterhin die typischen Abnahmen, welche das Ende der jeweiligen Phase definieren. Aktuell ist die Version 1.0 des 2005 erschienenen *V-Modell XT*.

Das hier dargestellte Modell ist nur ein Auszug des umfangreichen V-Modell. Die gewählte Vorgehensweise für dieses „Softwareprojekt“ lässt sich in fünf Punkten zusammenfassen:

1. Anforderungsanalyse
2. Systementwurf
3. Programmierung und Implementierung
4. Integration (Modul- und Systemtests)
5. Inbetriebnahme und Wartung

3.1.2 Ist-Situation

Eine vergleichbare Anwendung bzw. ein Vorgänger des zu entwickelnden Portals ist nicht vorhanden. Messergebnisse werden über Pangaea (bzw. verschiedene Benutzeroberflächen im Web mit Zugriff auf die Arbeitsdatenbanken), Publikationsdaten über ePIC abgerufen. Das System wird dementsprechend von Grund auf neu geschaffen.

3.1.3 Funktionalitäten

Für die Meteorologen ist es wichtig, dass sie fachspezifische Daten suchen können, d.h. es müssen spezielle Suchfelder zur Verfügung gestellt werden, um diese Daten zu erhalten. Für die Weiterverarbeitung der (Mess-)Daten selbst muss vom System die Möglichkeit bereitgestellt werden, sich diese als universell verwendbare Dateiformate (XML, CSV[1] etc.) herunterzuladen.

3.1.4 Berechtigte Nutzer

Benutzt werden kann das Portal von jeder Person, welche mit dem Internet verbunden ist. Eine Authentifizierungssystem ist für den Prototypen nicht vorgesehen. Für einige, unveröffentlichte Pangaea-Daten ist eine Mitgliedschaft in dem jeweiligen Projekt und damit ein Login erforderlich, dies wird jedoch aus rechtlichen Gründen vorerst über Pangaea selbst abgewickelt.

3.1.5 Verständlichkeit

Das Portal muss so einfach und unkompliziert wie möglich gehalten werden. Eine Anlehnung an das Aussehen etablierter Suchmaschinen oder Portale ist anzustreben. Die dargebotenen Funktionen müssen weitestgehend selbsterklärend sein,

[1] Character Separated Values

weitere Informationen, Hilfestellungen und Problemlösungen sollten auf einer separaten Hilfeseite erläutert werden.

3.1.6 Wart- und Erweiterbarkeit

Der Quelltext ist verständlich zu halten und muss sinnvoll dokumentiert werden. Die Struktur ist so zu wählen, das Änderungen leicht durchzuführen sind. Objektorientierte Programmierung ist – wenn möglich – vorzuziehen.

3.1.7 Internationalisierung

Das Portal wird von vornherein in englischer Sprache verfasst. Die Nutzer können, auch ohne profunde Kenntnisse dieser Sprache, durch die bereits erwähnte Anlehnung an das Aussehen gängiger Suchmaschinen die Formularfelder des Portals intuitiv nutzen.

3.1.8 Standardisierung

Bei der Entwicklung sollen verschiedene Standards des W3C verwendet werden, damit die Anwendung aktuellen Standards entspricht. In nachfolgender Tabelle sind diese aufgeführt.

Name	**Version**	**Verweis**
Extensible Markup Language	1.1	[W3C XML 2004]
Web Service Description Language	1.2	[W3C WSDL 2003]
Simple Object Access Protocol	1.2	[W3C SOAP 2003]
Extensible Hypertext Markup Language	1.1	[W3C XHTML 2001]
Cascading Style Sheets	2.0	[W3C CSS 2005]

Tabelle 3.1: Verwendete W3C-Standards

3.2 Pflichtenheft

Dieses Pflichtenheft ist ein weiterer Schritt der Anforderungsanalyse. Zu beachten ist die ggf. notwendige Aufteilung der Punkte nach dem Skript, welches die Vorarbeit für das Portal leistet und dem Portal (sowie der dahinter befindlichen Anwendung) selbst. Die hier verwendeten Punkte erfolgten in Anlehnung an [Balzert 1999a, S. 464 - 467].

3.2.1 Zielbestimmung

3.2.1.1 Muss-Kriterien

Skript: Das Skript muss über Auswahlmöglichkeiten (z.B. Schiff und Station) verfügen; dementsprechend die Option nach Reise bzw. Jahr, von welchem die Messdaten exportiert werden sollen. Es sollte sich – sofern möglich – modular erweitern lassen, um Abfragen auf andere meteorologische Arbeitsdatenbanken sowie sonstige Erweiterungen zu ermöglichen. Nach Abarbeitung des Skripts muss das Ergebnis als CSV-Datei bereitstehen.

Portal: Eine Aufteilung der Suchoptionen in „einfach“ und „erweitert“ (spezielle Suchfunktionen nach meteorologischen Kriterien) ist eine primäre Funktion. Die gefundenen Ergebnisse müssen zwischen Publikationen und Datensätzen getrennt dargestellt werden. Bei den Datensätzen (Messdaten) gehört die Darstellung der Daten hinzu, sowie eine Möglichkeit zum herunterladen. Alle Daten müssen korrekt, vollständig und wie in ihrem Ursprungssystem gespeichert zur Darstellung gebracht werden. Eine Seite mit Hilfestellungen für die Benutzer ist ebenfalls Pflicht.

3.2.1.2 Kann-Kriterien

Skript: Für die verfügbaren Reisen sollte eine Auflistung erfolgen. Da kein direkter Import der Daten in Pangaea erfolgen darf, sollte das Skript aus Komfortgründen eine Möglichkeit zum Versenden der erzeugten Datei an den Datenkurator beinhalten.

Portal: Die Messdaten sollten als XML-Datei zur Verfügung gestellt werden. Ein Navigator für die Suchergebnisse sowie die Anzeige der Anzahl der Ergebnisse können implementiert werden. Eine Möglichkeit, die Anzeige der Publikationen und Datensätze zu wechseln, d.h. das jeweils nicht benötigte auszublenden ist als hilfreich zu erachten. Die Anzeige einer Kurzfassung (Abstract) der Publikationen, sofern vorhanden sollte der Vollständigkeit halber eingefügt werden. Für die Versorgung des Benutzers mit aktuellen Informationen können eine Nachrichtenseite / ein Nachrichtendienst (RSS) angelegt werden.

3.2.1.3 Abgrenzungskriterien

Skript: Mit dem Auftraggeber wurde sich verständigt, im Rahmen der Diplomarbeit eine Beschränkung auf die Daten der synoptischen Observation vorzunehmen.

Das Skript wird dementsprechend angepasst. Weitere Anpassungen sowie andere Daten erfolgen außerhalb der Arbeit.

Portal: Der Prototyp benötigt in diesem Stadium keine Visualisierung und keinen selektiven Zugriff auf die in Pangaea gespeicherten Messdaten.

3.2.2 Einsatz

3.2.2.1 Anwendungsbereiche

Hauptanwendungsbereich ist die Meteorologie sowie ihr nahe stehende / verwandte Fachbereiche.

3.2.2.2 Zielgruppen

Skript: Das Skript wird nur von dem aktuellen Betreuer der meteorologischen Arbeitsdatenbanken verwendet und nach seinen Wünschen angepasst. Bei der Zielperson handelt es sich um einen Nicht-Informatiker mit erweiterten Kenntnissen der Programmierung.

Portal: Die Zielgruppe sind Meteorologen sowie an dieser Wissenschaft interessierte Personengruppen.

3.2.2.3 Betriebsbedingungen

Skript: Das Skript ist nur für den Gebrauch innerhalb des AWI-Netzwerkes bestimmt.

Portal: Es existieren keine besonderen Bedingungen, das Portal kann überall, wo eine Verbindung zum Internet besteht und eine Workstation, ein Desktop-PC etc. verfügbar ist, aufgerufen werden. Die Handhabung erfolgt wie die Benutzung einer „normalen“ Internetseite, sprich ohne Aufsicht und besondere Betriebszeit.

3.2.3 Umgebung

3.2.3.1 Software

Skript: Eine lauffähige Perl-Umgebung (Ver. 5) unter Sun Solaris (ab Ver. 8) ist notwendig. Ferner müssen Schreibrechte des Benutzers im Ausführungsverzeichnis des Skripts vorhanden sein.

Portal: Die Implementierung des Portals erfolgt auf einem Apache WebServer (Ver. 2.0.54) in Verbindung mit PHP 5 (Ver. 5.0.4). Als Betriebssystem für den Server ist Sun Solaris 9 im Einsatz. Schnittstellen existieren mittels SOAP zu dem Informationssystem Pangaea und dem Repositorium Fedora (Ver. 2).

3.2.3.2 Hardware

Skript: Eine Sun Workstation (derzeit Sun Ultra 5) oder eine Verbindung per Terminal (via Telnet etc.) auf den entsprechenden Server wird benötigt.

Portal: Als Server ist eine Sun Fire 15K vorgesehen. Weitere Hardware wird nicht benötigt. Auf die Hardware bzw. Plattformen des Benutzers kann kein Einfluss genommen werden, unterschiedlichste Kombinationen sind möglich. Meist wird es sich um einen handelsüblichen PC, MAC oder eine Workstation handeln.

3.2.3.3 Orgware

Portal: Es muss Rücksprache mit dem Datenkurator genommen werden, um das Format der Exportdatei, welche das Skript liefern soll, festzulegen.

Skript: Vor dem Einsatz des Portals müssen die meteorologischen Daten aus einer Arbeitsdatenbank nach Pangaea exportiert werden. Dies soll mittels Skript realisiert werden. Ebenfalls gilt es das Fedora-System mit den Publikationsdaten zu füllen.

3.2.4 Funktionalität

Skript: Nach dem Aufrufen muss der Benutzer eine Auswahl darüber erhalten, welchen Datenstamm er exportieren möchte. Es erfolgt die Abarbeitung seiner Auswahl.

Portal: Der Benutzer sucht mittels einfacher oder erweiterter Suchoption nach Publikationen bzw. Daten. Nach Ausgabe und Auflistung der Ergebnisse kann er diese sich im Detail anschauen, oder falls es mehr als Zehn (in einer Ergebnisgruppe) sind, weitere anzeigen lassen. Im Detail hat der Benutzer die Möglichkeit (bei Datensätzen) sich die Messdaten anzusehen oder als CSV und XML-Datei herunterzuladen.

3.2.5 Daten

Skript: Behandelt werden nur meteorologische Daten ohne besondere Metadaten.

Portal: Es handelt sich um Veröffentlichungen mit und ohne meteorologischen Dateninhalten. Insgesamt stehen zusammen in beiden System über 260.000 Einträge zum Abruf bereit.

3.2.6 Leistungen

Skript: Es sind keine besonderen Leistungsvoraussetzungen notwendig.

Portal: Die Verfügbarkeit der verwendeten Web Services muss innerhalb von zehn Sekunden bestätigt werden. Andernfalls wird eine Fehlermeldung geworfen. Aufgrund der unterschiedlichen Größe der Publikationen, welche Messdaten enthalten, deren Aufrufsgeschwindigkeit von der Schnelligkeit der Internetverbindung bzw. Reaktion des hinter dem Web Service stehenden Servers abhängt, werden keine weiteren Leistungsangaben gemacht. Der Benutzer wird auf die Größe und ggf. auf die Dauer für das Anzeigen bzw. Herunterladen der Messdaten hingewiesen.

3.2.7 Benutzungsoberfläche

Skript: Es handelt sich um ein typisches Unix-Shellskript, welches in der Konsole abgearbeitet wird. Von daher ist auf eine verständliche Menüstruktur zu achten.

Portal: Die Oberfläche soll sich an das „Corporate Design“ des Alfred-Wegener-Instituts anlehnen. Das Alter der Zielgruppe beginnt bei ca. 20 Jahren und ist nach oben offen. Dementsprechend muss auf die Ergonomie des Portals geachtet werden.

3.2.8 Qualitätsziele

Skript: Das Skript muss für einen Nicht-Informatiker mit erweiterteten Programmierkenntnissen wartbar und erweiterbar sein.

Portal: Wichtig für das Portal sind eine hohe Benutzerfreundlichkeit sowie eine hohe Änderbarkeit. Des Weiteren ist auf die Wartbarkeit zu achten, da die Anwendung ggf. entsprechend verschiedener Änderungen und Anpassungen der angebundenen Systeme aktualisiert werden muss.

4 Vorbereitung der Prototypentwicklung – Datentransport

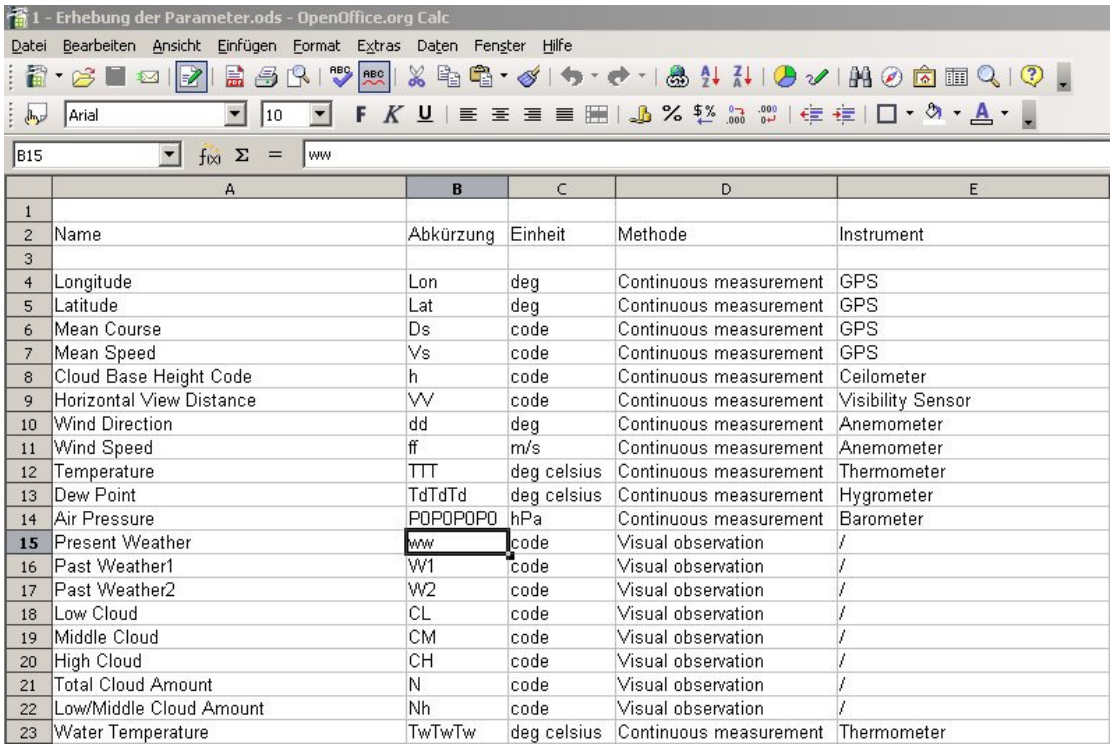
4.1 Erfassung der zu verarbeitenden meteorologischen Daten

Vor der Entwicklung des Portals steht ein umfassender Datentransport. Wie bereits erwähnt befinden sich die meteorologischen Messdaten in einer Arbeitsdatenbank. Von dort müssen sie nach Pangaea exportiert werden, um bei einer entsprechenden Abfrage über das Portal angezeigt zu werden.

Vor der Entwicklung des dafür notwendigen Skripts sind einige Vorbereitungen notwendig:

1. Rücksprache mit dem Datenkurator
2. Erstellung einer Liste mit den benötigten Parametern
3. Verteilung der Parameter-ID's

Bei den Treffen mit dem Datenkurator wurde die Vorgehensweise für den Transport der Daten festgelegt. An erster Stelle stand die Erstellung einer Liste mit den wissenschaftlichen Parametern, welche nach Pangaea aufgenommen werden sollten. Diese Liste (s. Abb. 4.1) enthält neben den Parameternamen auch deren Abkürzung und Einheit. Des Weiteren sind hier den Parametern die Methoden hinzugefügt, mit der die jeweiligen Daten erhoben wurden und, sofern verfügbar, das jeweils verwendete Messinstrument.



	A	B	C	D	E
1					
2	Name	Abkürzung	Einheit	Methode	Instrument
3					
4	Longitude	Lon	deg	Continuous measurement	GPS
5	Latitude	Lat	deg	Continuous measurement	GPS
6	Mean Course	Ds	code	Continuous measurement	GPS
7	Mean Speed	Vs	code	Continuous measurement	GPS
8	Cloud Base Height Code	h	code	Continuous measurement	Ceilometer
9	Horizontal View Distance	VV	code	Continuous measurement	Visibility Sensor
10	Wind Direction	dd	deg	Continuous measurement	Anemometer
11	Wind Speed	ff	m/s	Continuous measurement	Anemometer
12	Temperature	TTT	deg celsius	Continuous measurement	Thermometer
13	Dew Point	TdTdTd	deg celsius	Continuous measurement	Hygrometer
14	Air Pressure	P0P0P0P0	hPa	Continuous measurement	Barometer
15	Present Weather	ww	code	Visual observation	/
16	Past Weather1	W1	code	Visual observation	/
17	Past Weather2	W2	code	Visual observation	/
18	Low Cloud	CL	code	Visual observation	/
19	Middle Cloud	CM	code	Visual observation	/
20	High Cloud	CH	code	Visual observation	/
21	Total Cloud Amount	N	code	Visual observation	/
22	Low/Middle Cloud Amount	Nh	code	Visual observation	/
23	Water Temperature	TwTwTw	deg celsius	Continuous measurement	Thermometer

Abbildung 4.1: Auszug Parameterliste

Diese Liste wurde nun in das Pangaea-System eingepflegt. Jedem Parameter wurde dabei eine eindeutige numerische ID zugeordnet. Bei einigen Parametern war das nicht mehr nötig, da sie bereits durch die Verwendung in anderen Projekten und Messreihen mit einer ID ausgestattet sind. Damit waren die Vorbereitungen nun abgeschlossen und die Entwicklung des Skripts konnte beginnen.

4.2 Praktische Umsetzung

Nach Abschluss der Vorbereitungen dieser Phase der Diplomarbeit beginnt die erste Programmierung. Diese Arbeit ist in den nachfolgenden Untersektionen erläutert.

4.2.1 Entwicklung

Zu Entwicklungsbeginn stand die Frage nach der passenden Sprache. In Betracht gezogen wurden mehrere Sprachen, übrig blieben Perl, C und Fortran. Das sind jene Sprachen, von denen der Verantwortliche für die meteorologischen Daten, welcher später mit dem Programm arbeiten soll, Kenntnisse besitzt. Dies ist insofern wichtig, wenn später Anpassungen vorgenommen werden müssen, weil sich z.B.

die Datenbankstruktur ändert. Daher ist auf einen verständlichen (gut dokumentierten) und leicht anpassbaren Quelltext zu achten.

Nach einigen Überlegungen fiel die Wahl auf die Skriptsprache Perl. C und Fortran sind zwar weitaus mächtiger, aber für das Ziel der Entwicklung überdimensioniert.

Das Skript ist eingeteilt in die Teile *Header*, *Body* und *Footer*. Der Header enthält die verwendeten Bibliotheken, Variablendeklarationen sowie die Programmanweisung für die korrekte Verarbeitung von Benutzereingaben innerhalb der verwendeten Konsole. Folgende Bibliotheken wurden eingebunden:

- Term::ReadKey
  - aktiviert Verarbeitung von Benutzereingaben
- Sybase::DBlib
  - ermöglicht Zugriff auf Sybase-Datenbanken
- Fcntl
  - *File Control System* zur Verarbeitung von Dateien
- MIME::Lite
  - erlaubt Mailversand per Skript, auch mit Anhängen

Der Body enthält die Programmanweisungen, welche für die Verarbeitung verantwortlich sind, der Footer die dafür verwendeten Funktionen. Diese Anweisungen wurden zur Dokumentation in Gruppen zusammengefasst und sind auf den folgenden Seiten erklärt.

4.2.1.1 Auswahl des Datenursprungs

Im ersten Menü kann der Benutzer auswählen, welchen Datenursprung die zu exportierenden Daten haben sollen. Abbildung 4.2 zeigt diese Auswahl. Es stehen Daten der Polarstern sowie Neumayer- und Koldewey-Station zu Verfügung. Die Auswahl erfolgt über Eingabe einer Zahl zwischen 1 und 3.

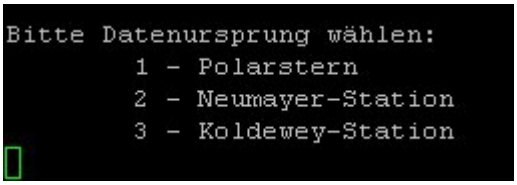


Abbildung 4.2: Exportskript: Menü, Datenursprung

Nun folgen je nach Auswahl unterschiedliche Menüs. Das Menü für Polarstern benötigt die Eingabe des Reiseziels, der Fahrtnummer sowie des Fahrtabschnitts (s. Abb. 4.3). Die beiden Stationen hingegen brauchen vom Benutzer einen Zeitraum, in welchem die Daten abgefragt werden (s. Abb. 4.4).

```
Bitte Reiseziel eingeben (z.B. ANT oder ARK)
ANT
Bitte Reisenummer eingeben (z.B. XXII)
XX
Bitte Fahrtabschnitt eingeben (z.B. 1 oder 4b)
2
```

Abbildung 4.3: Exportskript: Menü, Reisedaten (Polarstern)

```
Bitte Anfangszeit eingeben (z.B. 2007-01-01 00:00)
2001-01-01 00:00
Bitte Endzeit eingeben (z.B. 2007-12-31 21:00)
2001-12-31 21:00
```

Abbildung 4.4: Exportskript: Menü, Zeitraum (Stationen)

4.2.1.2 Eingabe der Benutzerdaten für die Datenbank

Wurden die Eigenschaften der Abfrage eingegeben folgt die Abfrage der Benutzerdaten für die Datenbank. Aus Gründen der Sicherheit wird das Passwort bei der Eingabe nicht auf dem Bildschirm ausgegeben, sondern unterdrückt (s. Abb. 4.5).

```
Geben Sie bitte Ihren Benutzernamen ein:
bbraeuer
Geben Sie bitte Ihr Benutzerpasswort ein:
```

Abbildung 4.5: Exportskript: Abfrage Benutzerdaten

4.2.1.3 Datenbankabfragen

Waren die Benutzerdaten korrekt, wird eine Verbindung mit der besagten Arbeitsdatenbank aufgebaut. Es folgt die Ausführung einer mit den eingegebenen Werten vordefinierten SQL-Abfrage. Das Ergebnis wird in einem Array gespeichert und Zeile für Zeile, tabulatorgetrennt in die Datei geschrieben. Sind die Benutzerdaten hingegen falsch, springt das Skript zurück zur Dateneingabe (s. Abb. 4.6).

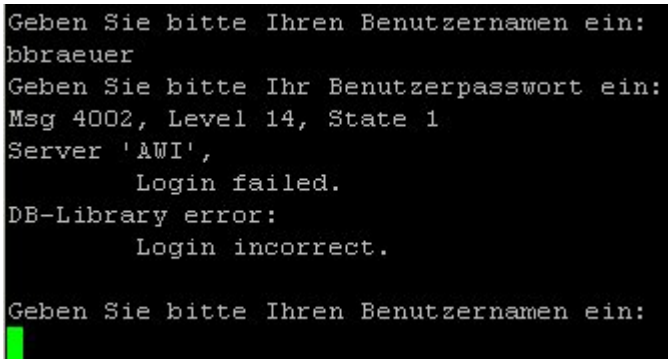


Abbildung 4.6: Exportskript: Abfrage Benutzerdaten Fehlerbehandlung

4.2.1.4 Dateioperationen

Der Name der zu schreibenden Datei setzt sich aus einem Präfix (exp-), dem Kürzel des Datenursprungs sowie den eingegeben Informationen zusammen, z.B.:

- exp-PS.ANT-XX-1
- exp-NM.2004_2005
- exp-NA.1999-01-01_2003-12-31

Bevor die Datei angelegt und Werte in sie eingetragen werden, überprüft das Programm das Vorhandensein einer Datei mit dem gleichen Namen. Der Benutzer erhält die Möglichkeit, diese Datei zu löschen oder die Datei weiter zu nutzen. Letzteres ist nicht empfehlenswert, da es ggf. für den Datenkurator beim Importieren zu Problemen führen kann. Die Abfrage ist daher eher eine Sicherheit für den Benutzer, damit er eine vorhandene Datei nicht unwissentlich überschreibt.

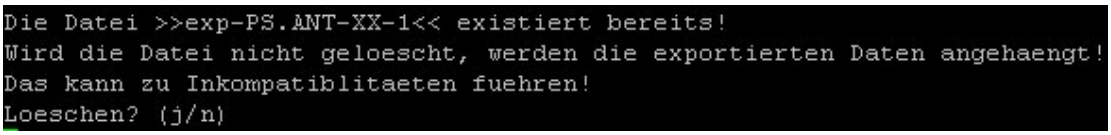


Abbildung 4.7: Exportskript: Abfrage existierende Datei löschen

Handelt es sich bei dem Datenursprung um Stationsdaten, wird die erzeugte Datei nach dem Eintrag „\tJ\t“ durchsucht. Wird das Skript fündig, ersetzt es diesen Eintrag mit „\tY\t“. Hintergrund dieser Operation ist die englische Bezeichnungsweise in Pangaea. Ist in der Arbeitsdatenbank ein Eintrag mit J(a) gekennzeichnet, muss er in Pangaea mit Y(es) gespeichert werden. Diese Operation passiert im Hintergrund, ohne Einwirken des Benutzers.

4.2.1.5 Mail-Versand

Ist die Datei erzeugt und mit den exportierten Werten gefüllt worden, hat der Benutzer nun die Möglichkeit, diese Datei zu verschicken. Es reicht die Eingabe der Empfängeradresse, Nachrichtenbetreff und -text sind im Quelltext festgelegt (die Einträge können dort ggf. einfach geändert werden). Die Mail wird nach der Eingabe zusammen mit der Datei als Anhang an die Zieladresse geschickt. Bei erfolgreichem Versand (s. Abb. 4.8) wird dieser bestätigt, tritt hingegen ein Fehler auf, beginnt die Mailprozedur erneut. Soll die Datei nicht versendet werden, ist das Programm an dieser Stelle beendet.

```
Soll die erstellte Datei als Mail verschickt werden? (j/n)
j
Wie lautet die Adresse des Empfaengers? (z.B. hgrobe@awi-bremerhaven.de)
bbraeuer@awi-bremerhaven.de
Mail an bbraeuer@awi-bremerhaven.de verschickt!
```

Abbildung 4.8: Exportskript: Erfolgreicher Mailversand

4.2.2 Ablauf

Um den Programmablauf besser nachvollziehen zu können, ist dieser in den Abbildungen 4.9 bis 4.12 als ereignisgesteuerte Prozesskette (EPK) dargestellt. EPK's werden in erster Linie eingesetzt, um Arbeitsprozesse graphisch darzustellen. Sie eignen sich, wie hier verwendet, für die Darstellung von Abläufen bei Programmen.

Symbol	Beschreibung
	*Ereignisse* sind Voraussetzung von Funktionen, können aber auch das Resultat von diesen sein
	*Funktionen* sind Darstellungen von Aktionen; sie werden durch ausgelöst und resultieren in diese
	*Disjunktion* / „oder"-Verknüpfung
	*Startereignis* / *Endereignis*

Tabelle 4.1: EPK-Symbole und deren Beschreibung

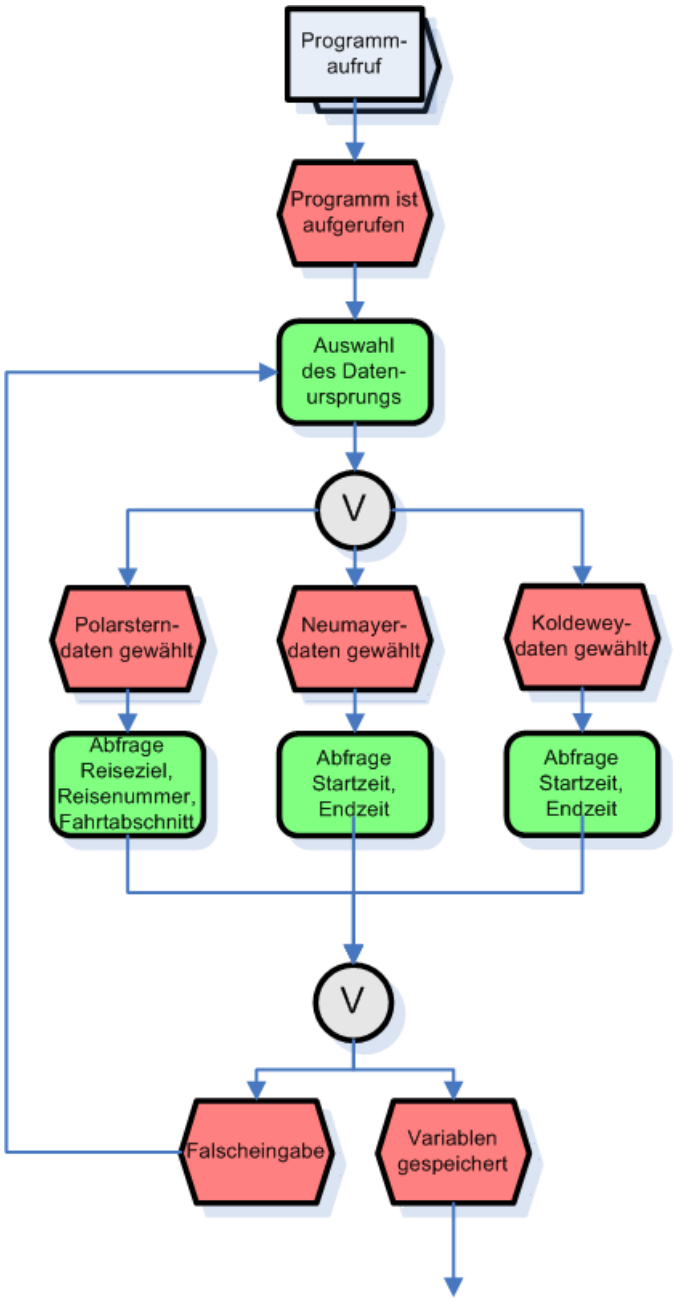


Abbildung 4.9: EPK des Exportskripts, Teil 1

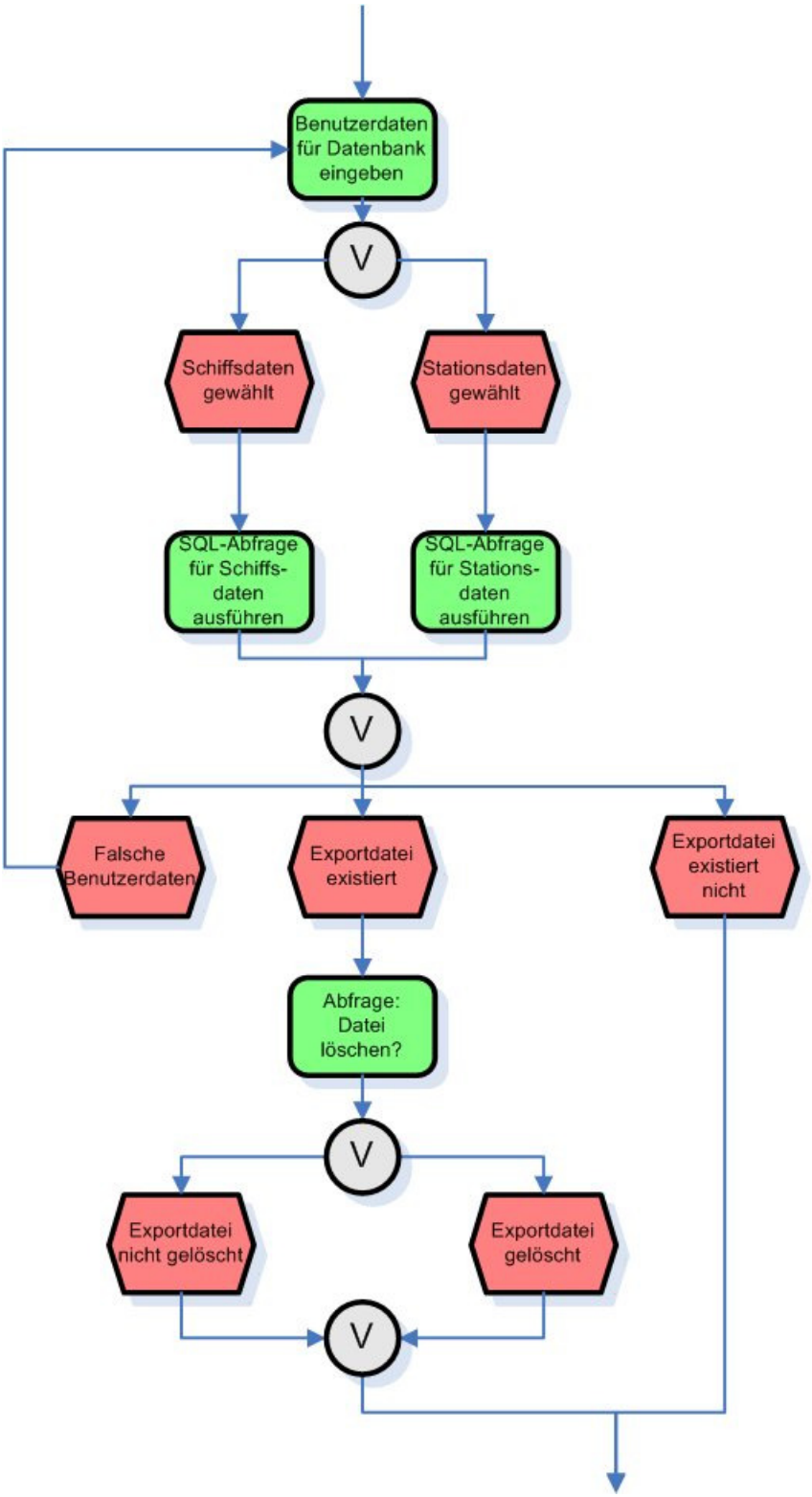


Abbildung 4.10: EPK des Exportskripts, Teil 2

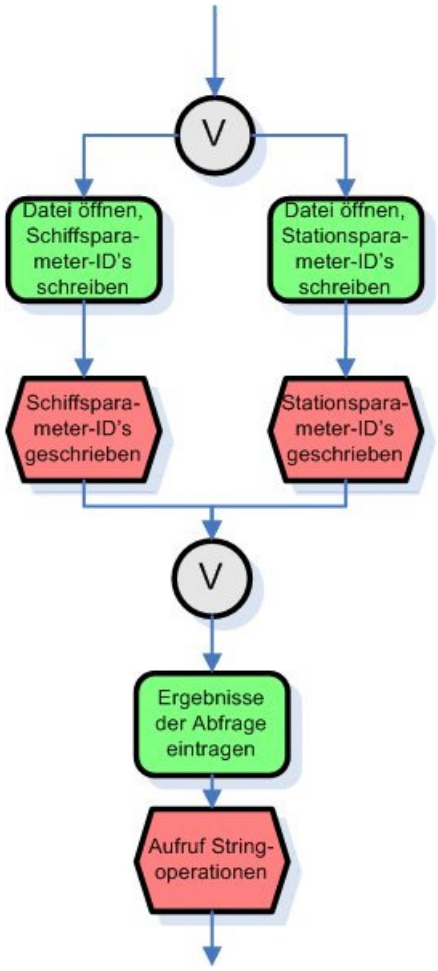


Abbildung 4.11: EPK des Exportskripts, Teil 3

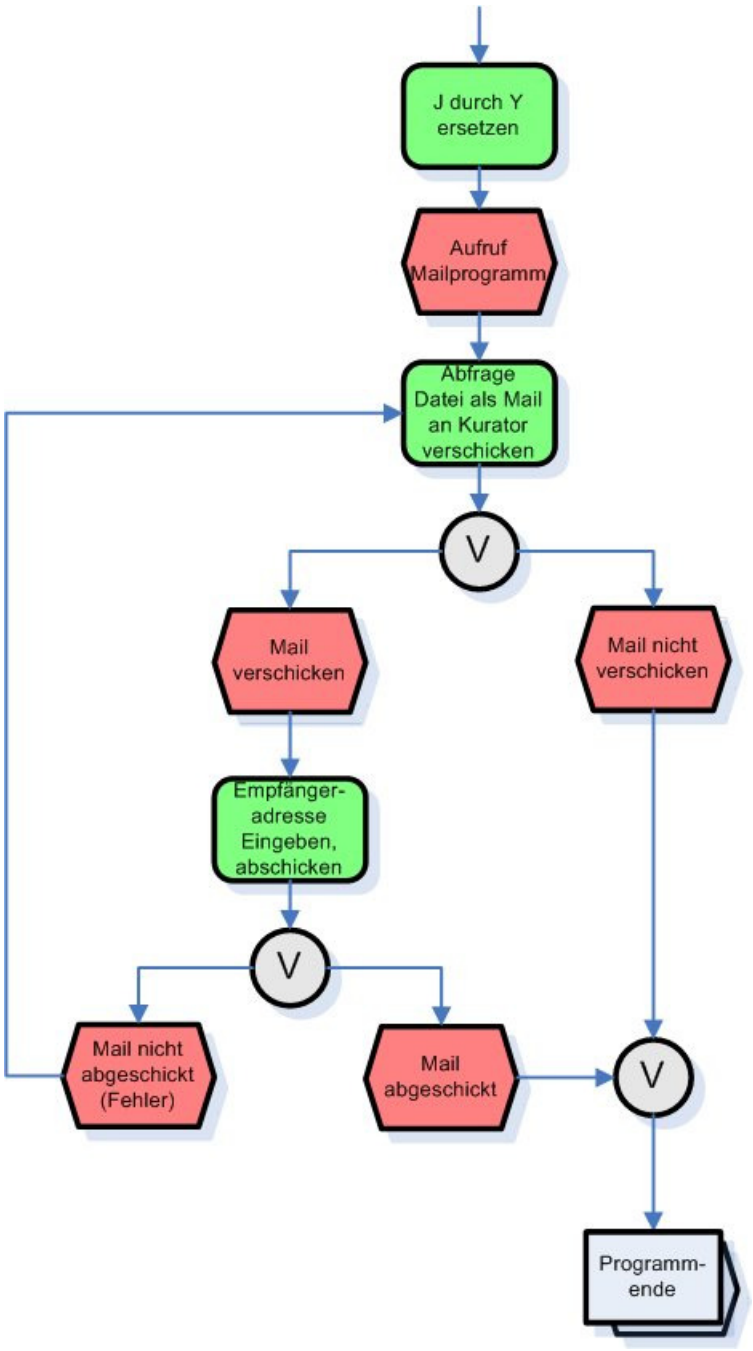


Abbildung 4.12: EPK des Exportskripts, Teil 4

4.2.3 Handreichung

Da das Skript als von dem Portal unabhängige Applikation zu betrachten ist, wird dafür eine separate Anleitung benötigt. Diese wurde in zwei Teile gegliedert. Der erste Teil befasst sich mit der grundlegenden Bedienung, der zweite ist als Hilfestellung für mögliche Anpassungsprogrammierungen zu betrachten, welche außerhalb der Diplomarbeit anfallen könnten.

4.2.3.1 Anleitung zur Bedienung

Das Programm befindet sich im Pfad */mobs1/user1/misawi/pangaea/* und trägt den Namen *export.pl*. Zum Starten des Skripts reicht die Eingabe dieses Namens in der Konsole, sofern man sich im genannten Pfad befindet. Danach gilt es sich an die Menüanweisungen zu halten. Die einzelnen Menüs wurden bereits unter dem Punkt 4.2.1 auf Seite 40ff. erläutert, von daher wird hier auf eine bebilderte Erklärung verzichtet.

Zu Beginn steht die Auswahl des Datenursprungs, durch Eingabe und Bestätigung einer der Zahlen 1, 2 oder 3 werden die entsprechenden Untermenüs aufgerufen. Das Polarsternmenü verlangt folgende Eingaben:

- Reiseziel → mögliche Werte:
  - *ANT* für Antarktis
  - *ARK* für Arktis
- Reisenummer → mögliche Werte:
  - in der Datenbank vorhandenen Reisen in Form von römischen Zahlen, als I, XV, XXII etc.
- Fahrtabschnitt → mögliche Werte:
  - in der Datenbank vorhandenen Fahrtabschnitte in der Form 1, 3, 5a etc.

Das Menü für der Stationen benötigt folgende Werte:

- Anfangszeit → mögliche Werte:
  - Datum nach ISO 8601, also z.B. 2004, 2003-06-01, 1999-02-01 21:00

- Endzeit → mögliche Werte:
  - Datum nach ISO 8601, also z.B. 2004, 2003-06-01, 1999-02-01 21:00

Wurden die jeweiligen Werte eingegeben, erscheint das Menü für die Benutzerdaten. Hier sind die Daten für den Zugang zum AWI-Datenbanksystem einzugeben. Ein Benutzerkonto in diesem System sowie mindestens Leserechte in der abzufragenden Datenbank sind Voraussetzung. Nun folgen, sofern die Daten richtig waren und die Datenbank erreichbar ist (vgl. Fehlermeldung der Datenbank in Abb. 4.6), die Abfragen. Diese erfolgen intern und können an dieser Stelle vom Benutzer nicht mehr gesteuert werden.

Zum Abschluss erfolgt die Frage nach dem Mailversand. Wird diese Option gewählt, fordert das Programm auf, die Emailadresse einzugeben, an welche die Datei geschickt werden soll. Nach erfolgreichem Versand ist das Programm beendet. Bei einem Misserfolg springt das Skript wieder zu der Frage, ob eine Mail verschickt werden soll. Bei einem „Nein“ an dieser Stelle ist das Programm ebenfalls beendet.

Hinweis → Den Abbruch des Programms kann der Benutzer zu jeder Zeit mit dem Tastenkürzel STRG + C bzw. CTRL + C erzwingen.

Hier eine Zusammenfassung des Ablaufs:

1. Programm aufrufen, *export.pl* in */mobs1/user1/misawi/pangaea/*
2. Auswahl des Datenursprungs
3. Eingabe der Daten, welche exportiert werden sollen
   - Reisedaten für Polarstern
   - Zeitraum für Stationen
4. Eingabe der Datenbankbenutzerdaten
5. leere Datei wird angelegt, wenn sie bereits existiert, Frage zum Löschen dieser Datei beantworten
6. Abfrage wird ausgeführt und das Ergebnis in Datei geschrieben.
7. Frage zum Mailversand beantworten, falls ja, Emailadresse des Empfängers eingeben.

4.2.3.2 Anleitung zum Einfügen weiterer Messeinrichtungen

Die Handhabung des Skripts wurde so einfach wie möglich gehalten. Dieses Prinzip wurde beim Quelltext fortgesetzt. Ziel dieses Teils der Handreichung ist das Einfügen einer neuen Messeinrichtung. Es dient der Vorlage für zukünftige Erweiterungen.

Arbeiten im Header Der Header enthält Daten, welche für das Skript benötigt werden (verwendete Bibliotheken, Variablendeklarationen etc.). Für die Erweiterung müssen hier **keine** Änderungen vorgenommen werden. Die nachfolgenden Hinweise dienen nur dem Verständnis.

Zwei Dinge sind im Header zu beachten, zum Einen das Einbinden weiterer Bibliotheken (z.B. SOAP-Paket, um Perl Web Service-fähig zu machen) (s. Quelltext 4.1) sowie das Setzen des Pfades (s. Quelltext 4.2), sollte Änderungsbedarf für diesen bestehen.

Hinter dem Befehl *use* ist der Name des entsprechenden Paketes bzw. der Bibliothek anzugeben. Bibliotheken, welche nicht im Perl-Kern enthalten sind, müssen sich in dem Pfad *use lib „/Pfad“* befinden.

```
1  use lib "/ordner/unterordner/unterunterordner";
2  use Bibliotheks- bzw. Paketname;
```

Quelltext 4.1: Exportskript: Bibliotheken einbinden

Der Pfad für das Skript ist in der Variable *$export_dat_pfad* gespeichert. Sollte die Anwendung verschoben werden etc. muss hier der neue Pfad eingegeben werden. Das Programm an sich funktioniert zwar von jeder Stelle aus, die Mailfunktion hingegen benötigt den Pfad für das Anhängen der Exportdatei.

```
1  $export_dat_pfad = "/mobs1/user1/misawi/pangaea/";
```

Quelltext 4.2: Exportskript: Pfad des Skripts ändern

Body Der Körper enthält die Hauptbestandteile des Skripts. Die größten „Blöcke“ wurden in separate Subroutinen (auch als Funktionen oder Methoden bekannt) gepackt und in den Footer des Skripts verschoben. Dies dient der Übersichtlichkeit und Wartbarkeit, da das Ändern separater Methoden leichter erfolgen kann als bei hintereinander geschriebenem Quelltext.

Als erstes ist das Startmenü zu erweitern (Auswahl des Datenursprungs). Hier werden mit dem *print*-Befehl die Namen der neuen Messeinrichtungen sowie einer Kennziffer hinzugefügt (s. Zeile 6-7 in Quelltext 4.3).

```
1  print "\nBitte Datenursprung wählen:\n";
2  print "\t1 - Polarstern\n";
3  print "\t2 - Neumayer-Station\n";
4  print "\t3 - Koldewey-Station\n";
5
6  #wird ergänzt mit:
7  print "\t4 - xyz-Einrichtung\n";
8  #usw.
9
10 $datenursprung = ReadLine 0;
11 chop $datenursprung;
```

Quelltext 4.3: Exportskript: Datenursprung bearbeiten

Da die Hauptprogrammteile in Subroutinen ausgelagert wurden, müssen im Body kaum Anpassungen vorgenommen werden. In der *while*-Schleife, welche kurz nach dem Befehl zum Ausführen der Datenbankabfrage (*$export->dbsqlexec*) folgt, gilt es nur den Erfolgstext (s. Quelltext 4.4, Zeile 16ff.) anzupassen. Dieser Teil des Programms dient der Benutzerinformation und ist für den korrekten Ablauf nicht unbedingt erforderlich.

```
1  #Textausgabe nach Abschluss des Schreibens
2  if ($datenursprung eq "1")
3  {
4    print "Schreiben von <<$reiseziel-$reisenr/$fahrtabschnitt>>
5      - Daten erledigt!\n";
6  }
7  elsif ($datenursprung eq "2")
8  {
9    print "Schreiben von Neumayer-Daten erledigt!\n";
10 }
11 elsif ($datenursprung eq "3")
12 {
13   print "Schreiben von Koldewey-Daten erledigt!\n";
14 }
15
16 #wird ergänzt mit:
17 elsif ($datenursprung eq "4")
18 {
19   print "Schreiben von xyz-Daten erledigt!\n";
20 }
21 #usw.
```

Quelltext 4.4: Exportskript: Erfolgstext bearbeiten

Für das Mailprogramm gilt ähnliches, hier muss der variable Teil des Nachrichtentextes (s. Quelltext 4.5, Zeile 14ff.) angepasst werden. Es wird wieder eine *elsif*-Kontroll-Struktur hinzugefügt. Zu Beachten ist die Art der Messeinrichtung, damit die richtigen Variablen ausgewählt werden.

```
1  # Variabler Teil des Nachrichtentextes
2  if ($datenursprung eq "1")
3  {
4    $text = "die Polarsternreise $reiseziel-$reisenr/$fahrtabschnitt";
5  }
6  elsif ($datenursprung eq "2")
7  {
8    $text = "die Neumayer-Station von $zeitraum1 und $zeitraum2";
```

```
9   }
10  elsif ($datenursprung eq "3")
11  {
12    $text = "die Koldewey-Station von $zeitraum1 und $zeitraum2";
13  }
14  #wird ergänzt mit:
15  elsif ($datenursprung eq "4")
16  {
17    print "die xyz-Einrichtung von $zeitraum1 und $zeitraum2!\n";
18  }
19  #usw.
```

Quelltext 4.5: Exportskript: Nachrichtentext bearbeiten

Footer Im Footer des Skripts existieren vier Subroutinen, von denen zwei angepasst werden müssen – *sub datenursprung* und *sub dbabfragen*. Anders als im Body sind Änderungen hier mit Vorsicht zu behandeln, da es um die Daten direkt geht. Ein Fehler, welcher es unbemerkt bis zum Import nach Pangaea schafft, könnte schwerwiegende Folgen für die Güte der Daten haben. Von daher sollten nach der Änderung und Ausführung des Programms die exportierten Daten genauestens überprüft werden!

Zuerst erfolgt der Aufruf der Subroutine *datenursprung($datenurpsrung)*. Als Parameter wird dieser Funktion die im Auswahlmenü des Datenursprungs eingegebene Zahl übermittelt. Ergänzt wird hier ein weiterer *elsif*-Block. Dieser Block setzt zu Beginn einige Variablen fest, welche später für die SQL-Anweisung benötigt werden. In diesem Beispiel (s. Quelltext 4.6) wird die *Messort_ID#* (Variable *$messort*) mit „99“, der Stationsname (Variable *$sname*) mit „xyz-Einrichtung“ und die Stationshöhe (Variable *$shoehe*) mit „22“ festgelegt. Danach erfolgen die notwendigen *ReadLine*-Befehle, um die Eingaben des Benutzers einzulesen und in einer Variable zu speichern.

```
1  sub datenursprung
2  {
3    ...
4    #Koldewey
5    elsif ($datenursprung eq "3")
6    {
7      $messort = "(3)";
8      $sname = "Koldewey";
9      $shoehe = "11";
10
11     print "Bitte Anfangszeit eingeben (z.B. 2007-01-01 00:00)\n";
12     $zeitraum1 = ReadLine 0;
13     chop $zeitraum1;
14
15     print "Bitte Endzeit eingeben (z.B. 2007-12-31 21:00)\n";
16     $zeitraum2 = ReadLine 0;
17     chop $zeitraum2;
18   }
19
20   #mögliche Erweiterung einer Station
21   elsif ($datenursprung eq "4")
22   {
```

```
23      $messort = "(99)";
24      $sname = "xyz-Einrichtung";
25      $shoehe = "22";
26
27      print "Bitte Anfangszeit eingeben (z.B. 2007-01-01 00:00)\n";
28      $zeitraum1 = ReadLine 0;
29      chop $zeitraum1;
30
31      print "Bitte Endzeit eingeben (z.B. 2007-12-31 21:00)\n";
32      $zeitraum2 = ReadLine 0;
33      chop $zeitraum2;
34    }
35
36    #... alle anderen Eingaben
37    else
38    {
39      goto DATENURSPRUNGSAUSWAHL;
40    }
41  }
```

Quelltext 4.6: Exportskript: Subroutine „datenursprung“ anpassen

Nun folgt die Routine *dbabfragen($datenursprung)*. Diese wird wie sonst auch mit einem *elsif*-Block ergänzt.

Der Block beginnt mit dem Aufruf der SQL-Abfrage. Es wird angeraten, die Abfrage zuvor in einem Datenbankwerkzeug zu erstellen und zu testen. So können die Ergebnisse der Abfrage geprüft und Fehler gleich erkannt werden. Als Werkzeug empfiehlt sich „SQL Advantage“, welches bei der Datenbankverwaltung „Sybase Central“ dabei ist. Es gibt drei Besonderheiten, denen Beachtung geschenkt werden muss.

Erstens: Für einige Werte der Abfrage tauchen Variablen auf. Diese sind vorher im Skript gesetzt worden, z.B. in der Subroutine *datenursprung*. Durch diese Variablen kann die Abfrage flexibel gehalten werden, was der Wartbarkeit zu Gute kommt.

Zweitens: Daten, die in Feldern des Typs *Float* gespeichert sind, müssen bei der Abfrage konvertiert werden, um ein einheitliches Bild und korrekt gerundete Werte zu erhalten. Das lässt sich mittels *convert(numeric(Gesamtstellenzahl, davon Anzahl der Dezimalstellen), Feldname)* bewerkstelligen (s. Quelltext 4.7).[1]

```
1 convert(numeric(5,1), Temperatur)
```

Quelltext 4.7: SQL-Beispiel: „convert“ 1

Damit das Pangaea-Importpogramm das Datum einlesen kann, muss es ebenfalls konvertiert werden. Hier hilft abermals die *convert*-Funktion (s. Quelltext 4.8). Hier wird die Länge des Datumstrings auf 26 Zeichen vom Typ „Character“ gesetzt sowie das Datumsformat „109“ festgelegt[2].

[1] vgl. [Sybase 1999, S. 8-12f]
[2] mon dd yyyy hh:mm:sss(AM oder PM), vgl. [Sybase 1999, S. 10-46f]

```
convert (char(26), DatumUhrzeit, 109)
```

Quelltext 4.8: SQL-Beispiel: „convert“ 2

Drittens: Hinter dem *select*-Befehl folgen die Spaltennamen, welche ausgewählt werden sollen. Auffallend ist hier z.B. „Wolkenuntergrenze *as '45259'*“. Für den Export hat der *as*-Befehl keine Bedeutung. In SQL-Advantage oder einem vergleichbaren Werkzeug ausgeführt verändert er jedoch die Spaltenbezeichnung. Bei den mehrstelligen Zahlen handelt es sich um die Pangaea-ID's. Dieser Befehl wurde in der Abfrage gelassen, um die Übersicht, welche Spalte welche ID hat, zu verbessern.

Nachdem die Abfrage ausgeführt wurde, speichert Perl die Ergebnisse in der Variable *$export*. Als nächstes wird eine Datei angelegt. Dabei wird mittels der Subroutine *fileexist($datei)* überprüft, ob die Datei bereits existiert und falls ja, ob sie gelöscht werden soll. Nun müssen noch ggf. die Pangaea-ID's angepasst werden. Hierbei gilt wieder: größtmögliche Vorsicht walten lassen! Eine vertauschte, fehlende oder falsche ID kann die wissenschaftliche Qualität der Daten negativ beeinflussen.

```
sub dbabfragen
{
  ...
  elsif ($datenursprung eq "2" or $datenursprung eq "3")
  {...}
  #wird ergänzt mit:
  elsif ($datenursprung eq "4")
  {
    $export->dbcmd("select  '$sname' as 'Event label',
                    '$shoehe' as '4607',
                    convert (char(26), DatumUhrzeit, 109) as '1599',
                    Wolkenuntergrenze as '45259',
                    HorSicht as '45260',
                    Windrichtung as '2221',
                    convert(numeric(5,1), Windgeschw) as '18906',
                    convert(numeric(5,1), Temperatur) as '4610',
                    convert(numeric(5,1), Taupunkt) as '4611',
                    convert(numeric(5,1), Luftdruck) as '2224',
                    ArtLuftdruckAenderung as '45311',
                    convert(numeric(5,1), BetragLuftdruckAenderung) as '45312',
                    GegenwWetter as '45261',
                    VergWetter1 as '45262',
                    VergWetter2 as '45263',
                    TiefeWolken as '45264',
                    MittlereWolken as '45265',
                    HoheWolken as '45266',
                    GesamtBedeckung as '45267',
                    BedeckungClCm as '45268',
                    convert(numeric(5,1), MaxTemperatur) as '5151',
                    convert(numeric(5,1), MinTemperatur) as '5150',
                    GegenwSchneetreiben as '45307',
                    VergSchneetreiben as '45308',
                    Whiteout as '45309'
            from  MetDB.dbo.ObseDat
            where Messort_ID# in $messort
            and   DatumUhrzeit between '$zeitraum1' and '$zeitraum2'
            order by DatumUhrzeit");
```

```
38      #Zerlege Datum und Uhrzeit, speichere in Array
39      @zeitraum1 = split(/ /, $zeitraum1);
40      @zeitraum2 = split(/ /, $zeitraum2);
41
42      # Anlegen der Exportdatei
43      $datei = "exp-XYZ.@zeitraum1[0]_@zeitraum2[0]";
44
45      #Funktionsaufruf
46      fileexist($datei);
47      open(DATEI,">>$datei");
48
49      # Pangaea-ID's in die Datei schreiben: VORSICHT - Stets mit größter Sorgfalt
            behandeln!
50      print DATEI "Event label\t4607\t1599\t45259\t45260\t2221\t18906\t4610\t
        4611\t2224\t45311\t45312\t";
52      print DATEI "45261\t45262\t45263\t45264\t45265\t45266\t45267\t45268
        \t5151\t5150\t45307\t45308\t45309\n";
54    }
55    # Rechte setzen (664 = Lese- und Schreibrechte fuer Eigner und Gruppe, sowie
          Leserechte fuer alle)
56    chmod(0664,$datei);
57  }
```

Quelltext 4.9: Exportskript: Subroutine „dbabfragen“ anpassen

Nun ist das Einfügen einer neuen Station abgeschlossen. Die Anpassungen sind nicht allzu komplex, sondern können mit Hilfe dieser Anleitung einfach und sicher durchgeführt werden. Zum Abschluss noch eine Übersicht über die wichtigsten Variablen.

Variablenname	**Beschreibung**
$srv	Setzt Datenbank-Server (AWI-intern)
$uid	Benutzername für die Datenbank
$pwd	Passwort für die Datenbank
$export_dat_pfad	Der Pfad der Exportdatei, meist auch der Ort, wo sich das Skript export.pl befindet
$datei	Name der Export-Datei
$datenursprung	Der Ursprung der Daten; Zahlen 1, 2, 3 usw.
$export	Variable für die Datenbankabfrage
$messort	Der Messort, Messort_ID#
$hoehe	Höhe der Messungen von Polarstern
$reiseziel	Reiseziel der Polarsternfahrt (ANT oder ARK)
$reisenr	Reisenummer der Polarsternfahrt
$fahrtabschnitt	Fahrtabschnitt der Polarsternfahrt
$sname	Der Stationsname (Neumayer, Koldewey)
$shoehe	Die Höhe der Station
$zeitraum1	Zeitraum der exportierten Daten, Beginn
$zeitraum2	Zeitraum der exportierten Daten, Ende

Tabelle 4.2: Exportskript: Übersicht der wichtigsten Variablen

5 MISAWIsta Suchportal – Prototypentwicklung

5.1 Erläuterung

Mit dem Transport der Daten aus der Arbeitsdatenbank nach Pangaea ist der erste Teil der Arbeit abgeschlossen. Der nächste Abschnitt ist die Entwicklung des Portal-Prototypen. Das Projekt trägt mit dem Entwicklungsbeginn den Namen „MISAWIsta“, wobei MIS für „Meteorology Information System“ steht.

Die Idee hinter dem Portal ist die Verbindung der Daten mehrerer Institutionen über die Web Service-Technologie. Der Benutzer bekommt eine Plattform gestellt, mit der er auf meteorologische Datenbestände zugreifen kann, ohne die abgefragten Systeme kennen zu müssen. Das Portal vereinigt die Suche in diesen Datenbeständen unter einer Oberfläche.

Im Alfred-Wegener-Institut läuft zur Zeit ein ähnliches Projekt, welches allerdings nach dem „Harvester“-Prinzip verfährt. Beim *harvesten* (engl. für: „ernten“) werden alle Daten aus den benötigten Systemen (welche natürlich entsprechende Schnittstellen bereitstellen und dieses Verfahren zulassen müssen) in eine Datenbank geschrieben und können lokal abgerufen werden.

Der Harvester überprüft dann z.B. einmal täglich die Systeme nach neuen Einträgen und kopiert sie in die besagte Datenbank. Der Vorteil dieses Verfahrens: Man ist nicht auf „fremde“ Systeme angewiesen und kann Oberflächen und andere Applikationen in ein eigenes System aufsetzen. Der Nachteil: Es müssen unnötigerweise eigene Ressourcen verbraucht werden, welche dann an anderer Stelle fehlen.

Durch die Nutzung bereitgestellter Web Services hingegen reichen ein paar hundert Kilobyte an Quelltexten, um auf die gewünschten Daten zuzugreifen. Die Vorteile dieses Verfahrens gegenüber einem Harvester[1]:

- stets die aktuellsten Daten
- fällt ein Dienst aus, bleiben die anderen davon unbetroffen

[1] für weitere Details s. [Onken 2005]

- keine Verschwendung eigener IT-Ressourcen, Abfragen etc. werden von der EDV der Service-Provider erledigt
- dadurch erhöht sich wiederum die Leistung des Web Service-Client

Aber auch die Nachteile dürfen nicht verschwiegen werden:

- Abhängigkeit von fremden Systemen
- damit verbunden die Verwendung unterschiedlicher Standards bei Datenhaltung, -bereitstellung etc.
- unter Umständen mangelhafter bis gar kein Support

Inzwischen haben sich diese Nachteile aber relativiert. Kein Institut oder Unternehmen, kann es sich leisten, eine Möglichkeit des Datenzugriffs zu veröffentlichen und diese dann „verkommen“ zu lassen. Von daher gibt es meistens auch eine gute Unterstützung oder zumindest eine Dokumentation der Web Services. Durch die fortschreitende Globalisierung und internationale Kooperation im wirtschaftlichen wie im wissenschaftlichen Sektor wird die Art der Daten- (bzw. Dienst-) Bereitstellung zunehmend vereinheitlicht. Zwar möchten verschiedene Interessengruppen ihr jeweiliges System als Standard durchsetzen, aber dennoch bieten diese dann meistens Schnittstellen zu anderen Standards.

Dem Endanwender, welcher den Client nutzt, ist es letztendlich egal, was hinter der Oberfläche passiert. Wichtig ist, dass er nach einer Aktion die gewünschte Reaktion hervorrufen kann, also in diesem Fall die Suche eines Begriffs und die Anzeige der gefundenen Daten. Ähnlich verhält es sich mit dem Entwickler, der einen Web Service nutzen will. Es ist ihm gleich, was für ein System hinter dem Service steht, wie die Bearbeitung der Anfrage funktioniert etc. Was er will, ist nach dem Aufrufen eines Web Services und der Parameterübergabe ein zurückgeliefertes Ergebnis, mit dem er arbeiten kann. Das ist ein großer Vorteil dieser Architektur. Der bereitgestellte Dienst hinter dem Web Service kann in einem gewissen Maß beliebig geändert werden, solange sich das Ergebnis des Aufrufes nicht ändert, bekommt der Entwickler von diesen Änderungen nichts mit und muss seinen Quellcode nicht anfassen. Alle Informationen, die ein Programmierer benötigt, erhält dieser theoretisch aus der Beschreibung, der WSDL. Natürlich können diese Beschreibungen mehr oder weniger aussagekräftig sein, so dass ggf. eine Dokumentation zu Rate gezogen werden sollte. Die technischen Details für die Kommunikation mit dem Web Service und dessen Nutzung sind darin aber vollständig beschrieben.

Fakt ist, die Nutzung von Web Services spielt eine immer größer werdende Rolle in der IT-Branche. Institutionen, die frühzeitig auf diesen „Zug“ aufspringen und in

die Entwicklung dieser Technologie investieren, werden früher davon profitieren, als Unternehmen, die sich dieser Innovation verschließen.

Doch zurück zum eigentlichen Thema. Für den Portal-Prototypen werden vorerst zwei Web Services verwendet, einer stammt dabei aus dem AWI selbst, der Andere ist extern bereitgestellt. Der Prototyp soll beweisen, dass der Einsatz dieser Technologie sinnvoll ist und die an sie gestellte Aufgabe der Datensuche effizient lösen kann. Hält man sich an die Vorgaben des V-Modells erfolgt nun, da der Punkt der Anforderungsanalyse abgeschlossen ist, der Entwurf des Systems.

5.2 Systementwurf

Der Entwurf enthält die grundlegendsten Regeln der Entwicklung des Portals.

5.2.1 Wahl der Sprache

Für die Entwicklung des Portals standen einige Sprachen zur Auswahl. Wie aus dem Kapitel Grundlagen bereits zu erahnen ist, fiel die Wahl auf PHP in der Version 5.0.4. PHP erfüllt für diese Aufgabe die notwendigen Voraussetzungen:

- Verarbeitung von Web Services
- Verarbeitung von XML-Dokumenten
- objektorientiert
- verminderter Pflegeaufwand

Mächtigere Sprachen, wie Java oder C/C++/C# kamen grundsätzlich auch in Frage, nur ist bei diesen (nicht allein aufgrund der höheren Typenstrenge) der Wartungsaufwand meist höher. Da Software-Entwicklung im AWI zum größten Teil ausgelagert ist, fehlen entsprechende Programmierer bzw. die vorhandenen sind mit anderen Projekten ausgelastet. PHP hingegen lässt sich einfacher erlernen und damit auch warten. Des Weiteren ist die Sprache für dynamische Webanwendungen ausgelegt. Dazu ein aktuelles Zitat:

> „PHP ist für dieses Umfeld entworfen worden. Es handelt sich eben nicht um eine Allzwecksprache, sondern sie ist webspezifisch. Die Art und Weise der Ausführung des Codes fällt deshalb nicht so stark ins Gewicht.“ – Andi Gutmans, Vice President Technology von Zend Technologies; In: [Graser 2005]

5.2.2 Corporate Design

Da das Portal in erster Linie von außen erreicht werden soll, muss es auch ansprechend gestaltet werden. Daher wurden für diese Arbeit einige Regeln aufgestellt. Oben, in der Kopfzeile, ist ein Banner abgebildet. Wichtig dabei sind die Maße von max. 700 Pixeln Breite und 110 Pixeln Höhe. Neben einem gestalterischen Aspekt ist gerade die Breite wichtig für die Ergonomie, welche im nächsten Punkt behandelt wird. Das zur Zeit verwendete Banner zeigt die Abbildung 5.1.

Abbildung 5.1: Banner

Der Hintergrund ist Weiß zu halten. Hyperlinks bzw. Verweise werden in dem AWI-Blau[2] dargestellt. Anderer Text soll wie üblich in schwarz dargestellt sein. Sonstige Elemente (wie z.B. Formulare) *können* in hellem Grau[3] gehalten werden.

Weitere Vorgaben wurden nicht gemacht. Die restliche Gestaltung kann frei gewählt werden.

5.2.3 Softwareergonomische Aspekte

Im Pflichtenheft wurde bereits erwähnt, dass die Klientel des Portals in jedwedem Alter, also vom jungen Meteorologiestudenten bis hin zum hochbetagten Wissenschaftler, sein kann. Dementsprechend ist das Portal zu gestalten, um nicht eine Gruppe auszuschließen.

Das menschliche Auge liest (digitale) Texte am Besten, wenn sie in der Form schwarze Schrift auf weißem Grund verfasst sind. Alternativ darf der Hintergrund auch in hellem Grau sein. Die Schrift sollte daher zwischen 10 und 12 Punkten liegen, und einer serifenlosen Form vorliegen, z.B. Verdana, Arial oder Helvetica. Serifenschriften sind zwar in gedruckter Form besser lesbar, im Webbereich bei kurzen Texten ist eine serifenlose Schrift allerdings vorzuziehen.

Das Portal wurde unter einer Auflösung von 1280 * 1024 Bildpunkten entworfen. Durch die festgelegte Breite des Banners ist auch für Benutzer mit niedriger Auflösung (bis 800 * 600) kein Problem, das Portal ohne Darstellungseinschränkungen oder -fehler zu nutzen.

[2]Hex-Farbcode #006ba5
[3]Hex-Farbcode #ebebeb

Formularfelder, Menüs und Buttons sind selbsterklärend gestaltet, damit auch jemand, der nicht der englischen Sprache mächtig ist, das Portal für seine Suche verwenden kann. Die Begleittexte wurden ebenfalls kurz und prägnant gehalten.

Für einen barrierefreien Zugang für Menschen mit verschiedenen Behinderungen ist das Portal konform zu den WAI[4]-Kriterien (sowie den darauf basierenden *Section 508*-Richtlinien für webbasierte Informationen und Anwendungen – N° 1194.22[5]) zu halten. Erklärtes Ziel des W3C ist es, das World Wide Web möglichst vielen Menschen zugänglich zu machen. Daher wird versucht, so viele Kriterien[6] wie möglich zu erfüllen. Angestrebt ist dabei als Minimum die *Level Double-A Conformance to Web Content Accessibility Guidelines 1.0*, die zweite von drei Stufen der WAI.

Ein Beispiel für diesen barrierefreien Zugang ist die bei der Entwicklung verwendete Gestaltung im „tableless design“. Dies bedeutet, das keine Tabellen zur Gestaltung der Seite genommen werden, was leider immer noch zu häufig gemacht wird. Screenreader, also Software zum Vorlesen von Bildschirminhalt für z.B. blinde Menschen, haben teilweise große Probleme mit einem auf Tabellen basierenden Design. Dank der vielfältigen Möglichkeiten von CSS und den sog. div-Containern (div = division = Bereich) kann aber auf Tabellen für die *Gestaltung* gänzlich verzichtet werden. Des Weiteren vermindert sich dadurch auch der Umfang des Quelltextes, was wiederum das Transfervolumen senkt. Die Tabellen können somit für das verwendet werden, wofür sie gedacht sind: zur Präsentation von Daten.

5.2.4 Entwurf einer Systemstruktur

Bevor mit der Programmierung begonnen wird, sollte anhand der bisherigen, theoretischen Überlegungen eine Struktur des zu entwickelnden Systems entworfen werden. An dieser schematischen Darstellung (s. Abb. 5.2) können nun weitere Schritte geplant und umgesetzt werden. Der Ablauf umfasst die Verbindung zwischen den Gruppen Benutzer und Portal (Zugriff mit Browser über Intranet / Internet), Portal und Web Services (Verbindung über PHP und SOAP) sowie Web Services und der dahinter befindlichen Dienste (Verbindungsart irrelevant, üblicherweise Java, C/C++/C# etc.).

In der Abbildung wurde ein Web Service *X* mit aufgenommen. Dieser steht stellvertretend für die zukünftige Einbindung anderer Dienste. Die Pfeile mit der Doppelspitze kennzeichnen den Rückfluss der Informationen / Daten zwischen den einzelnen Gruppen.

[4] Web Accessibility Initiative
[5] vgl. http://www.section508.gov/index.cfm?FuseAction=Content&ID=12#Web
[6] vgl. http://www.w3c.de/Trans/WAI/webinhalt.html

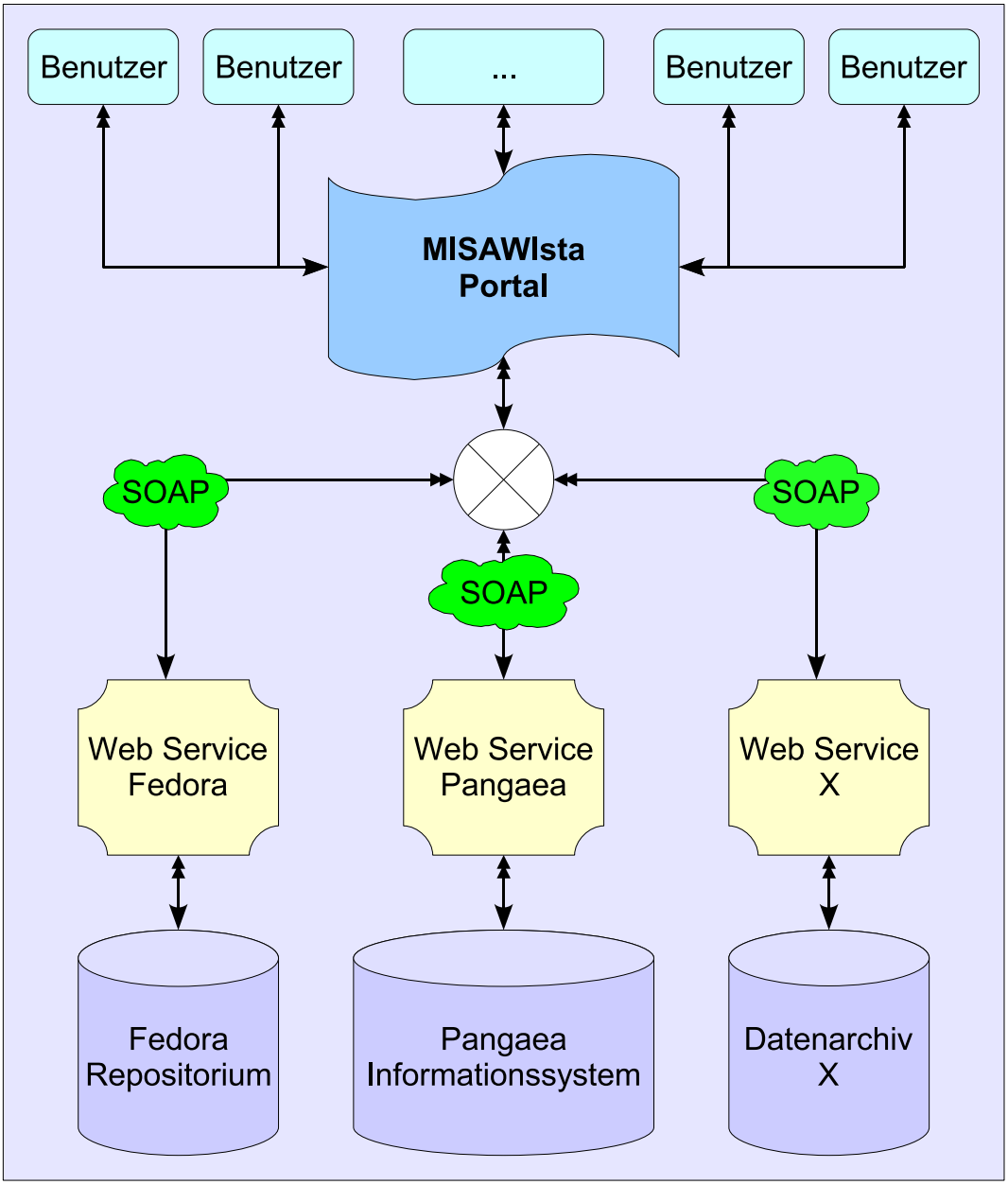


Abbildung 5.2: Systemstruktur

5.2.5 Entwurf einer Seitenstruktur

Wichtig ist die Beschränkung der angebotenen Seiten auf das Ziel, welches mit dem Portal erreicht werden soll. Es liegt daher im Interesse des Entwicklers, den Nutzern eine einfache Seitenstruktur anzubieten. Diese Struktur ist in Abb. 5.3 dargestellt.

Abbildung 5.3: Seitenstruktur

Von der Startseite aus, welche auch die einfache Suche (Free Search) enthält lassen sich über Hyperlinks fünf weitere Seiten erreichen: die erweiterte Suche (Advanced Search), Hilfe (Help), Verweise (Links), Neuigkeiten (News) sowie das Impressum (About). Als Ergebnis einer der beiden Suchen gelangt der Nutzer auf die Seite mit den aufgelisteten Ergebnissen (Results). Von diesen kann man sich die Details der Publikationen (Publication Details) oder Datensätze (Dataset Details) anschauen. Der Benutzer gelangt also (bei Verwendung der einfachen Suche) auf geradlinigem Wege zu seinem gewünschtem Ergebnis.

Die Tabelle 5.1 zeigt noch einmal aufrufbaren Seite mit einer Kurzbeschreibung. Genauere Informationen und Details zu den Seiten werden in den nachfolgenden Sektionen beschrieben.

Seite	**Beschreibung**
Startseite / Free Search	Die Startseite, enthält einfache Suche
Advanced Search	Erw. Suche, enthält spezielle Suchfunktionen
Links	Verweise auf die zugegriffenen Systeme
About	Impressum
Help	Hilfe für die Benutzung des Portals
Results	Aufgelistete Ergebnisse
Publication Details	Details einer Publikation
Dataset Details	Details eines Datensatzes

Tabelle 5.1: Portal: Erläuterungen zur Seitenstruktur

5.2.6 Beispielseite

Nach den verschiedenen strukturellen und gestalterischen Überlegungen wird es Zeit für ein erstes konkretes Beispiel. Zuvor jedoch zeigt die Abbildung 5.4 den durchgängig eingehaltenen Aufbau der Seiten selbst.

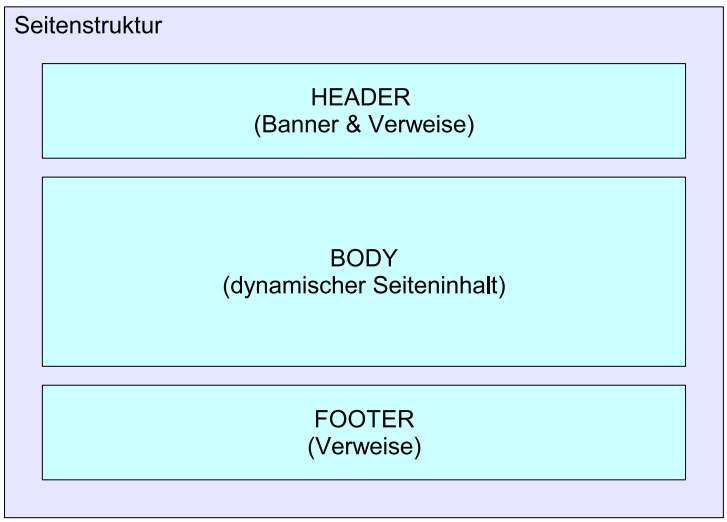


Abbildung 5.4: Aufteilung der Seite

Die Seite wurde in drei Teile gegliedert. Der *Header* bildet den *Kopf* des Portals. Er enthält, neben den auf der Seite selbst nicht sichtbaren (X)HTML-typischen Metadaten, das Banner. In der Mitte, im *Body* wird der eigentliche Inhalt der Seite, also Text, Formulare etc. abgebildet. Der *Footer* bzw. die Fußzeile beinhaltet verschiedene Verweise. Der Inhalt von Header und Footer ist statisch und wird auf jeder Seite eingebunden.

Die fertige Startseite (hier noch ohne Funktionsbeschreibung) zeigt Abb. 5.5. Zu erwähnen ist, dass die Abbildung hier so dargestellt wird, als wäre sie bei einer Auflösung von 1024 * 768 Bildpunkten gemacht. Es gibt keine falschen Umbrüche oder andere Darstellungsfehler, die Transformation der Seite verläuft in diesem Stadium fehlerfrei. Dieses Verhalten ist für den weiteren Projektverlauf als Standard anzustreben.

Abbildung 5.5: Startseite als fertiger Entwurf

5.3 Implementierung

Im Punkt Implementierung folgen Erläuterungen zur eigentlichen Entwicklung. Dabei wird nicht jede Zeile Quelltext durchgegangen – dafür sind die Kommentare im Code selbst da – sondern auf die Besonderheiten in der Programmierung, spezielle Funktionen und Abläufe etc. eingegangen. Die Struktur der Implementierungsbeschreibung hält sich an das Vorgehen bei der praktischen Entwicklung, dargestellt in nachfolgender Aufzählung. Die praktische Entwicklung wiederum wurde nach der Fertigstellung der Seitenstruktur (s. Abb. 5.4 S. 62ff) entworfen.

1. Vorbereitende Einstellungen
2. Entwicklung eines Grundgerüsts
3. Verarbeitung der Anfrage
4. Vorbereitung der Verbindung
5. Verbindung zu Pangaea

6. Verbindung zu Fedora
7. Verarbeitung der Daten
8. Gestaltung
9. Abschließende Entwicklung

5.3.1 Vorbereitende Einstellungen

Um alle gedachten Funktionen verwenden zu können, müssen in einige Einstellungen der PHP-Konfiguration des Apache Web Servers vorgenommen werden. Die Aktivierung der Erweiterungen erfolgte bei der Kompilierung von PHP 5.0.4. Es wurden die Optionen „*–with-sybase-ct=/opt/csw*“ (Zugriff auf Sybase Datenbanksysteme), „*–enable-soap*“ (Verwendung von SOAP bzw. SOAP-basierenden Web Services), „*–with-iconv=/opt/csw*“ und „*–with-iconv-dir=/opt/csw*“ (Konvertierung in verschiedene Zeichenkodierungen), „*–enable-session*“, „*–enable-trans-sid*“ (Aktivierung von Sessions und Weitergabe der Session-ID) verwendet.

5.3.2 Entwicklung eines Grundgerüsts

5.3.2.1 Startseite index.php

Der Anfang der Programmierung wurde mit der Entwicklung der Startseite gemacht. Ohne die Verwendung von PHP wurde die Seite in XHTML entwickelt. Das Formular (Suchfeld) wurde dabei bereits mit eingebaut, aber es enthält noch keine Funktion. Das Ergebnis dieser Anfangsarbeit wurde bereits in Abb. 5.5 auf Seite 65 gezeigt.

Nun wird das Suchfeld mit Funktion hinterlegt. Wichtig hierbei ist die Übergabe der eingegeben Daten an die Datei search.php mit der GET-Methode. Diese Methode ist eine der von HTTP unterstützten Request-Methoden („Anfrage-Varianten“), um Dateien abzurufen oder - wie in diesem Beispiel - ein Formular abzusenden. Die Variablen werden bei der GET-Methode in der URL übergeben und dort „gespeichert“, d.h. sie stehen der Programmiersprache wieder zur Verfügung und können bei einer erneuten Abfrage wieder verwendet werden. Das Ergebnis ist ein einfaches Formular, dargestellt in Quelltext 5.1.

```
1 <form action="search.php" method="get">
2     <p>
3       <label for="searchfield">Search Field
4         <input type="text" tabindex="0" id="searchfield" name="query" size="50"
              alt="search field" value="<?
5         # displays an error-message if nothing was inserted in the field
6         if ( $_GET["em"] == "1" ) echo "please insert valid search term"; else
              echo $_GET["query"]; ?>" />
```

```
7       </label>
8       <!--some important variables for the search-->
9       <input type="hidden" name="coord" value="true" />
10      <input type="hidden" name="data" value="1" />
11      <input type="hidden" name="pub" value="1" />
12      <input type="hidden" name="ref" value="i" />
13      <input type="submit" value="Search" class="button" />
14    </p>
15 </form>
```

Quelltext 5.1: Portal: index.php Suchfeld 1

Die Daten der Suchanfrage sowie die Daten vom Typ *hidden* werden nun an *search.php* übergeben. Durch den *hidden*-Typ werden beim Benutzen des Buttons zusätzlich zu der Eingabe im Suchfeld für die spätere Verarbeitung benötigte Daten übergeben. Die Variablen *data* und *pub* „sagen“ dem Skript, dass auf der Ergebnisseite sowohl die Datensätze als auch Publikationen angezeigt werden sollen. Variable *ref* hingegen kennzeichnet den Ursprung der Anfrage.

5.3.2.2 Erweiterte Suche search_advanced.php

Die erweiterte Suche soll den meteorologisch versierten Benutzern eine genauere Suche bieten, z.B. nach den Ergebnissen einer bestimmten Polarstern-Reise. Für diesen Zweck stellt die erweiterte Suche drei Suchmöglichkeiten zur Verfügung.

Freie Suche Die freie Suche ist soweit identisch zur einfachen Suche, bietet aber die Möglichkeit, Koordinaten einzugeben. Die Suchanfrage beschränkt sich dann auf Daten im angegebenen Bereich. Dies gilt nicht für Publikationsdaten, da diese derzeit über keine Koordinatenangaben in den Metadaten verfügen.

Suche nach Schiffsdaten Die Suche nach Daten von „Research Vessels“, also Forschungsschiffen, erfolgt über zwei Auswahllisten. Die erste Liste enthält die Daten der Schiffe, sprich den Namen und (sofern verfügbar) die Auflistung der Reisen. Die zweite Liste enthält die Messmethoden.

Um die Listen zu füllen, existieren zwei Funktionen. Die erste Funktion *expedition()* füllt die Liste mit den Daten der Schiffe. Für die Polarstern greift sie zusätzlich auf eine Sybase-Datenbank zu, in welcher die Reisedaten des Schiffes gespeichert sind und gibt sämtliche Fahrtabschnitte aus. Die anderen Schiffe sind die „Heinke“ und die „Meteor“. Für diese Schiff sind keine weiteren Daten vorhanden.

Die zweite Funktion, welche den Namen *choose_measurement()* trägt, generiert die Liste mit den Messmethoden. Hierfür wird ihr der feste Wert *ship* übergeben.

Suche nach Stationsdaten Für die „in situ land based“ (die Stationen) gibt es drei Auswahllisten. Liste Eins enthält die Namen der Stationen, Liste Zwei die Jahre und Liste Drei die Messmethoden.

Jeder Liste ist wieder eine Funktion zugeordnet. Die Funktion *platform()* füllt die Liste mit den Stationsnamen. Für die Berechnung der Jahre ist die Funktion *yearcnt()* zuständig. Ihr wird der Wert *$stationyear* übergeben, der zur Zeit auf das Jahr 1981[7] festgesetzt ist. Die Jahre werden von diesem Zeitpunkt bis zum vorangegangen Jahr angegeben. Die dritte Funktion ist wieder *choose_measurement()*, die hier mit dem festen Wert *platform* aufgerufen wird.

5.3.2.3 Sonstige Anpassungen

Sessions Da mit vielen skriptübergreifenden Variablen zu rechnen ist (spätestens wenn weitere Web Services hinzugefügt werden), wird der Einsatz der Session-Technologie notwendig, um die Verwaltung der Variablen durchzuführen. Dabei erhält jeder Benutzer, der eine Suchanfrage stellt, eine eigene Identifikationsnummer, die sog. Session-ID. In der Session (Sitzung) werden dann die jeweiligen Variablen gespeichert und stehen dann bis zum Ablauf der Session oder ihrer „Zerstörung“ zur Verfügung.

Sessions gehören zum PHP-Kern und bedürfen keiner besonderen Installation. Um eine neue Session anzulegen, reicht der Aufruf der PHP-Funktion *session_start()* an der Stelle des Skripts, ab der man die Sessions benötigt. Dieser Aufruf muss aber in jedem Skript erfolgen, in dem man auf die Sessionvariablen zugreifen möchte. Dabei wird aber nur beim ersten Aufruf des Nutzers eine neue Session erzeugt. Der Aufruf ist in Quelltext 5.2 dargestellt.

```
1 # destroy all session-data
2 kill_session();
3
4 # start a new session
5 session_start();
```

Quelltext 5.2: Portal: Session

Vor dem Start der Session wird noch eine andere Funktion aufgerufen, nämlich *kill_session()*. Dies ist eine eigene Funktion, die dafür sorgt, dass beim Aufruf der Startseite oder der erweiterten Suche (und nur dort) eine evt. vorhandene Session vollständig gelöscht wird. So werden Komplikationen mit alten, noch gespeicherten Variablen vermieden. Die Funktion (s. Quelltext 5.3) löscht ein evt. vorhandenes Cookie, leert die $_SESSION-Variable und „zerstört“ die Sitzung.

```
1 function kill_session()
2 {
3   # clear cookie
```

[7] Jahr der ersten Aufzeichnung auf der Georg-von-Neumayer-Station (Neumayer I)

```
4   unset($_COOKIE[session_name()]);
5   # clear superglobal $_SESSION and its content
6   unset($_SESSION);
7   # destroy session data and prevent warnings
8   @session_destroy();
9 }
```

Quelltext 5.3: Portal: Funktion kill_session()

Buffer Durch die *Output Control*-Funktionen von PHP kann gesteuert werden, wann vom Skript Ausgaben erfolgen sollen. Wie in Quelltext 5.4 zu sehen, befindet sich ein *echo*-Befehl zwischen der *ob_start()* und der *ob_end_flush()*-Funktion. Der String „test“ wird in den internen Puffer geschrieben und erst ausgegeben, wenn der Aufruf von *ob_end_flush()* erfolgt. Durch das Schreiben in den Puffer und die „gesammelte“ Ausgabe des Pufferinhalts lässt sich gegenüber einer sofortigen Ausgabe die Leistung steigern.

```
1 ob_start();
2 echo "test";
3 ob_end_flush();
```

Quelltext 5.4: Portal: Buffer

5.3.3 Verarbeitung der Anfrage

Wurden die entsprechenden Eingaben vom Benutzer getätigt, erfolgt nun die Verarbeitung. Dies geschieht in dem Skript *search.php*. Dieser Teil der Anwendung lässt sich in vier Abschnitte gliedern. Im ersten Abschnitt werden die erwähnten Benutzereingaben überprüft und in entsprechende Variablen gespeichert. Es wird überprüft, ob sie nicht schon als Teil einer Session vorhanden sind. Neben dem Suchstring, welcher hier verarbeitet bzw. bei einer detaillierte Anfrage zusammengefügt wird, werden auch die Koordinaten sowie die Entscheidung, ob das Portal nur Datensätze (*$data*), nur Publikationen (*$pub*) oder beides anzeigen soll, abgehandelt.

Die Verarbeitung der Variablen wird beispielhaft an den Koordinaten erklärt. Der notwendige Code ist in Quelltext 5.5 abgebildet. Die Abfrage überprüft, ob der mit der GET-Methode gelieferte Wert der Variable „coord“ true ist. Falls ja, werden die in Zeile 3 bis 6 abgebildeten Werte als Sessionvariablen gespeichert. Ansonsten, falls nicht schon entsprechende Werte in der Session vorhanden sind, werden die wieder mittels GET-Methode übertragene Benutzerwerte verwendet.

```
1 # process coordinates
2 if ($_GET["coord"] == true)
3 {
4   $_SESSION["minLat"] = -90;
5   $_SESSION["minLon"] = -180;
6   $_SESSION["maxLat"] = 90;
```

```
7   $_SESSION["maxLon"] = 180;
8 }
9 else
10 {
11   if (!$_SESSION["minLat"]) $_SESSION["minLat"] = $_GET["minLat"];
12   if (!$_SESSION["minLon"]) $_SESSION["minLon"] = $_GET["minLon"];
13   if (!$_SESSION["maxLat"]) $_SESSION["maxLat"] = $_GET["maxLat"];
14   if (!$_SESSION["maxLon"]) $_SESSION["maxLon"] = $_GET["maxLon"];
15 }
```

Quelltext 5.5: Portal: Koordinatenverarbeitung

Für den Bereich, zwischen dem die angezeigten Ergebnisse liegen sollen, sorgt der so genannte „Offset“. Der Bereich liegt dann zwischen dem Wert der Variable *$offset* und dem Ergebnis von *$offset* + *$count*. In der Variable *$count* ist die Anzahl der angezeigten Ergebnisse pro Liste (nicht pro Seite!) gespeichert.

Der zweite Abschnitt beschäftigt sich mit dem Web Service von Pangaea, Abschnitt drei mit dem von Fedora. Diese beiden Programmteile werden in den nachfolgenden Punkten behandelt.

Der letzte Abschnitt sorgt für die Ausgabe auf dem Bildschirm. Hier werden die von den beiden Web Services gelieferten Daten verarbeitet und formatiert. Des Weiteren werden hier der Navigator (für das Navigieren zwischen den Ergebnisseiten) sowie die „Sidebar“ (Infos und Auswahl der angezeigten Listen) aufgerufen und ausgegeben.

5.3.4 Vorbereitung der Verbindung

Um die Möglichkeiten der Objektorientierung in PHP sinnvoll zu nutzen, wurde vor Beginn der Verarbeitung der Web Services eine Klasse *webservice* angelegt, in welcher die verwendeten Variablen mit dem Befehl *var* definiert werden. Alle anderen Klassen können diese Variablen nun „erben“. Bei den Variablen handelt es sich um *$wsdl* (URI der WSDL), *$client* (erstellter SOAP-Client) und *$error* (Fehlervariable), dargestellt in Quelltext 5.6.

```
1 class webservice
2 {
3   var $wsdl;
4   var $client;
5   var $error;
6 }
```

Quelltext 5.6: Portal: Klasse *webservice*

5.3.5 Verbindung zu Pangaea

Im Skript search.php wird ein neues Objekt angelegt (`$webservice_pangaea = new webservice_pangaea`). Über dieses Objekt werden nun die verschiedenen

Funktionen aufgerufen und verarbeitet. Diese Funktionen befinden sich in der Klasse *webservice_pangaea*. Diese erbt die Variablen der Klasse *webservice* (über Befehl *extends*) und besteht aus den vier Funktionen:

- webservice_pangaea()
  - generiert den SOAP-Client, über welchen auf die bereitgestellten Methoden des Web Service zugegriffen werden kann
- register()
  - fordert vom Pangaea-System eine interne Session an
- search_small($offset, $count)
  - liefert Ergebnisse mit grundlegenden Details zurück, benötigt für die Ergebnisliste von Pangaea
  - bekommt den Bereichsanfang *$offset* und die Anzahl der Ergebnisse pro Liste *$count* übergeben
- show_detail($doi)
  - liefert vollständige Details eines Ergebnisses zurück, benötigt für die Detailansicht
  - bekommt den DOI des anzuzeigenden Ergebnisses übergeben

Über den Aufruf dieser Funktionen erhält man nun die Daten des Pangaea-Web Service. Dieser hat die Besonderheit, vor der Abfrage der gesuchten Daten eine Session zu benötigen. Die Session ist für den internen Gebrauch in Pangaea vorgesehen und verhindert durch die Speicherung verschiedener Daten (z.B. Browsertyp) bei neuen Suchanfragen Leistungseinbußen. Von daher ist diese Session bei Anfragen mit zu übergeben. Da sie wie jede andere Session einmal abläuft (was schon eintreten kann, wenn der Benutzer noch bei der Recherche ist), wird in der *search.php* überprüft, ob die Pangaea-Session noch existiert oder ob eine neue angefordert werden muss. Hinweis: Die Pangaea-Session steht in keinem Zusammenhang mit der Session des Portals (außer das sie selbst in einer MISAWIsta-Session-Variable gespeichert wird).

Jede Funktion ist mit *try/catch*-Blöcken ausgestattet. Der auszuführende Code steht im *try*-Block. Wird bei der Ausführung ein Fehler festgestellt, werden eine *exception* (engl. für „Ausnahme“) geworfen und die Anweisungen im catch-Block ausgeführt. Somit können Probleme des Web Service erkannt und auf sie reagiert werden.

Die Ergebnisse der Funktionen *search_small()* und *show_detail()* werden über SOAP im XML-Format zurückgeliefert. PHP versteht diese Antwort als Objekt. Die Beschreibung der Verarbeitung erfolgt im übernächsten Punkt „Verarbeitung der Daten von Pangaea“. Die Quelltexte 5.7 und 5.8 zeigen die SOAP-Anfrage an den Web Service und dessen Antwort[8].

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://PanWebServices" xmlns:xsd="http://www.w3.org/2001/XMLSchema
    " 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:search>
      <session xsi:type="xsd:string">9ce22f44511ea3398410882dc8d8b7a6</session>
      <query xsi:type="xsd:string">kÃ¶nig neumayer 1985</query>
      <minLat xsi:type="xsd:double">-90</minLat>
      <minLon xsi:type="xsd:double">-180</minLon>
      <maxLat xsi:type="xsd:double">90</maxLat>
      <maxLon xsi:type="xsd:double">180</maxLon>
      <offset xsi:type="xsd:int">0</offset>
      <count xsi:type="xsd:int">10</count>
    </ns1:search>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quelltext 5.7: Portal: SOAP-Request (Pangaea)

```
<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:searchResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
        encoding/" xmlns:ns1="http://PanWebServices">
      <searchReturn href="#id0"/>
    </ns1:searchResponse>
    <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.
        xmlsoap.org/soap/encoding/" xsi:type="ns2:PangaVistaResult" xmlns:soapenc
        ="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns2="http://
        PanWebServices">
      <timeInDatabase href="#id1"/>
      <WFSBasePath xsi:type="soapenc:string">/ws/wfs/PangaVista/9
          ce22f44511ea3398410882dc8d8b7a6/567f50849622f98d6cb5520e2bd4c419?</
          WFSBasePath>
      <session xsi:type="soapenc:string">9ce22f44511ea3398410882dc8d8b7a6</
          session>
      <totalCount href="#id2"/>
      <lowerLimit href="#id3"/>
      <WFSCapabilitiesPath xsi:type="soapenc:string">/ws/wfs/PangaVista/9
          ce22f44511ea3398410882dc8d8b7a6/567f50849622f98d6cb5520e2bd4c419?
          SERVICE=WFS&amp;VERSION=1.0.0&amp;REQUEST=GetCapabilities</
          WFSCapabilitiesPath>
      <offset href="#id4"/>
      <results soapenc:arrayType="ns2:PangaVistaEntry[1]" xsi:type="ns2:
          ArrayOfPangaVistaEntry">
        <item href="#id5"/>
      </results>
    </multiRef>
```

[8] Der Wert des Elementes <xml> (PD94bWwg[...]dD4K) ist ein mehr als 2000 Zeichen langer Code (Base64), der aus Platzgründen mit [...] abgekürzt wurde.

```
18    <multiRef id="id5" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="ns3:PangaVistaEntry" xmlns:ns3="
          http://PanWebServices" xmlns:soapenc="http://schemas.xmlsoap.org/soap/
          encoding/">
19      <xml xsi:type="soapenc:base64">PD94bWwg[...]dD4K</xml>
20      <score href="#id6"/>
21    </multiRef>
22    <multiRef id="id2" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="xsd:int" xmlns:soapenc="http://
          schemas.xmlsoap.org/soap/encoding/">1</multiRef>
23    <multiRef id="id3" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="xsd:boolean" xmlns:soapenc="http://
          schemas.xmlsoap.org/soap/encoding/">false</multiRef>
24    <multiRef id="id4" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="xsd:int" xmlns:soapenc="http://
          schemas.xmlsoap.org/soap/encoding/">0</multiRef>
25    <multiRef id="id1" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="xsd:int" xmlns:soapenc="http://
          schemas.xmlsoap.org/soap/encoding/">244</multiRef>
26    <multiRef id="id6" soapenc:root="0" soapenv:encodingStyle="http://schemas.
          xmlsoap.org/soap/encoding/" xsi:type="xsd:byte" xmlns:soapenc="http://
          schemas.xmlsoap.org/soap/encoding/">82</multiRef>
27  </soapenv:Body>
28 </soapenv:Envelope>
```

Quelltext 5.8: Portal: SOAP-Response (Pangaea)

5.3.6 Verbindung zu Fedora

Die Verbindung zum Fedora-Web Service verläuft ähnlich. Ein neues Objekt wird angelegt (`$webservice_fedora = new webservice_fedora`) und die Funktionen aufgerufen. Die Klasse *webservice_fedora* erbt ebenso die Variablen der Klasse *webservice*. Folgende Funktionen sind implementiert:

- webservice_fedora()
  - generiert den SOAP-Client, über welchen auf die bereitgestellten Methoden des Web Service zugegriffen werden kann
- search_small($count)
  - die Suchabfrage; liefert die Ergebnisse der Suche zurück
  - bekommt die Anzahl der Ergebnisse pro Liste *$count* übergeben
- resume($fid)
  - setzt die Suche fort
  - bekommt Fedora Session übergeben

- show_detail($pid)
  - liefert vollständige Details eines Ergebnisses zurück, benötigt für die Detailansicht
  - bekommt den PID des anzuzeigenden Ergebnisses übergeben

Wie schon Pangaea arbeitet auch Fedora mit einer internen Session. Im Gegensatz zu Pangaea ist diese allerdings nicht für die interne Leistung des Systems zuständig, sondern für Fortführung der Suche. Fedora lässt sich nicht anhand von des „Offset“ steuern, sondern über die Session und das auch nur in Richtung „Vorwärts“. Das heißt, ohne gewisse Kniffe bekommt man immer nur die nächsten Ergebnisse der Suche, kann aber nicht auf die vorhergehenden zugreifen. Vom Prinzip her ist es sinnlos, für jede Suchanfrage (und sei sie auch zum gleichen Begriff) eine neue Session zu generieren. Dies hebelt das Sessionprinzip der längerfristigen Speicherung von generierten Daten aus. Damit muss aber vorerst gelebt werden, bis es vielleicht bei einer neuen Version von Fedora geändert wird.

Um eine Steuerung per „Offset“ zu ermöglichen, werden bei der Suchanfrage die größtmögliche Liste (derzeit max. 100 Einträge) angefordert und die Listeneinträge in ein Array gespeichert. Dieses Feld lässt sich nun beliebig verwenden und die darin befindlichen Ergebnisse ähnlich Pangaea steuern. Zu beachten ist aber, dass es sich dabei um eine Not- bzw. Übergangslösung handelt. Diese sollte daher bei einer Änderung des Fedora-Systems entsprechend angepasst werden.

Der Rückgabewert des Fedora-Web Service über SOAP im XML-Format ist ein Objekt mit verschiedenen Arrays, welche von der Programmiersprache direkt verarbeitet werden kann. Das Objekt besteht aus zwei Obereinträgen. Im ersten Eintrag, der *listSession* befindet sich die eben genannte Session sowie zusätzliche Informationen. Der zweite Eintrag, *resultList*, enthält die Ergebnisse der Suche. Die Quelltexte 5.9 und 5.10 zeigen die SOAP-Anfrage an den Web Service und dessen Antwort.

```
1  <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/">
2    <SOAP-ENV:Body>
3      <ns1:findObjects>
4        <resultFields SOAP-ENC:arrayType="xsd:string[16]" xsi:type="ns2:
             ArrayOfString">
5          <item xsi:type="xsd:string">title</item>
6          <item xsi:type="xsd:string">creator</item>
7          <item xsi:type="xsd:string">subject</item>
8          <item xsi:type="xsd:string">description</item>
9          <item xsi:type="xsd:string">publisher</item>
10         <item xsi:type="xsd:string">contributor</item>
11         <item xsi:type="xsd:string">date</item>
12         <item xsi:type="xsd:string">type</item>
```

```
13          <item xsi:type="xsd:string">format</item>
14          <item xsi:type="xsd:string">identifier</item>
15          <item xsi:type="xsd:string">source</item>
16          <item xsi:type="xsd:string">language</item>
17          <item xsi:type="xsd:string">relation</item>
18          <item xsi:type="xsd:string">coverage</item>
19          <item xsi:type="xsd:string">rights</item>
20          <item xsi:type="xsd:string">pid</item>
21        </resultFields>
22        <maxResults xsi:type="xsd:nonNegativeInteger">999</maxResults>
23        <query xsi:type="ns2:FieldSearchQuery">
24          <conditions xsi:nil="1"/>
25          <terms xsi:type="xsd:string">kÃ¶nig*</terms>
26        </query>
27      </ns1:findObjects>
28    </SOAP-ENV:Body>
29  </SOAP-ENV:Envelope>
```

Quelltext 5.9: Portal: SOAP-Request (Fedora)

```
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
             [77]">
10         ...
11         <item href="#id14"/>
12         <item href="#id15"/>
13         <item href="#id16"/>
14         <item href="#id17"/>
15         <item href="#id18"/>
16         ...
17       </resultList>
18     </multiRef>
19     ...
20     <multiRef id="id14" soapenc:root="0" soapenv:encodingStyle="http://schemas.
           xmlsoap.org/soap/encoding/" xsi:type="ns5:ObjectFields" xmlns:ns5="http
           ://www.fedora.info/definitions/1/0/types/" xmlns:soapenc="http://schemas.
           xmlsoap.org/soap/encoding/">
21       <pid xsi:type="xsd:string">awi:Kni2004e</pid>
22       <label xsi:type="xsd:string" xsi:nil="true"/>
23       <fType xsi:type="xsd:string" xsi:nil="true"/>
24
25       <cModel xsi:type="xsd:string" xsi:nil="true"/>
26       <state xsi:type="xsd:string" xsi:nil="true"/>
27       <ownerId xsi:type="xsd:string" xsi:nil="true"/>
28       <cDate xsi:type="xsd:string" xsi:nil="true"/>
29       <mDate xsi:type="xsd:string" xsi:nil="true"/>
30       <dcmDate xsi:type="xsd:string" xsi:nil="true"/>
31       <title xsi:type="xsd:string">Long-term monitoring of ozone profiles in
             Antarctica</title>
32       <creator xsi:type="xsd:string">KÃ¶nig-Langlo, G.</creator>
```

```
33
34        <creator xsi:type="xsd:string">Herber, A.</creator>
35        <subject xsi:type="xsd:string">Climate System; Physical and Chemical
              Processes in the Atmosphere</subject>
36        <subject xsi:type="xsd:string">EARTH SCIENCE &amp;gt; Atmosphere  &amp;gt;
              Atmospheric Chemistry; Neumayer, OMI &gt; Ozone Measuring Instrument,
              OMI-EOS &gt; Ozone Monitoring Instrument-EOS, OZONESONDES, POAM III &gt
              ; Polar Ozone and Aerosol Measurement III, TOMS &gt; Total Ozone
              Mapping Spectrometer</subject>
37
38        <description xsi:type="xsd:string">uri:http://www.awi-bremerhaven.de/php/
              PublicationAbstracts/abstract.php?uid=Kni2004e</description>
39        <publisher xsi:type="xsd:string">XXVIII SCAR Open Science Conference, 24 -
              31 July 2004, Bremen, Germany, Poster S16 / P10., 2004</publisher>
40        <date xsi:type="xsd:string">2004-01-01</date>
41        <type xsi:type="xsd:string">event, poster</type>
42        <identifier xsi:type="xsd:string">Kni2004e</identifier>
43        <identifier xsi:type="xsd:string">uri:http://www.awi-bremerhaven.de/php/
              PublicationAbstracts/abstract.php?uid=Kni2004e</identifier>
44
45        <identifier xsi:type="xsd:string">awi:Kni2004e</identifier>
46        <rights xsi:type="xsd:string">uri:http://creativecommons.org/licenses/by-nc
              -nd/2.0/legalcode</rights>
47      </multiRef>
48      ...
49    </soapenv:Body>
50  </soapenv:Envelope>
```

Quelltext 5.10: Portal: SOAP-Response (Fedora)

5.3.7 Verarbeitung der Daten

Vor der Verarbeitung wird in *search.php* wieder ein neues Objekt (`$listResult = new listResult`) angelegt. Die Klasse *listResult* enthält u.a. die Funktionen, mit denen die Daten von Pangaea und Fedora aufbereitet werden.

Die Detailansicht wird über das Skript *showdetails.php* aufgerufen. Je nachdem, woher das Ergebnis stammt (Fedora oder Pangaea) wird die entsprechende Datei aufgerufen, die für die Detailansicht verantwortlich ist.

5.3.7.1 Verarbeitung der Suchergebnisse von Pangaea

Die Ergebnisanzeige der Pangaeadaten wird über die Funktion *pangaea_res()* generiert. Ihr werden die Werte *$result_pangaea* (das vom Web Service gelieferte Ergebnis), *$pangaea_error* (Fehlerstatus) und *$metadata* (Adresse der Namensraumdefinitionen) übergeben.

Nun wird über eine Schleife jedes Ergebnis durchlaufen und entsprechend formatiert auf dem Bildschirm ausgegeben. Jedes Element muss aber vorher in drei Stufen bearbeitet werden, damit es verarbeitet werden kann.

1. base64_decode()

    - Die Ergebnisse kommen von Pangaea im Base64-Format[9] kodiert an. Normalerweise wird das Format von der Programmiersprache erkannt und automatisch umgewandelt, PHP 5 aber hat bei dieser Konvertierung noch einen Fehler, so dass die Umwandlung manuell (per Funktion) erfolgen muss. Der Fehler sollte in einer der nächsten Versionen von PHP behoben sein.

2. simplexml_load_string()

    - Wie erwähnt liegen die (nun dekodierten) Suchergebnisse im XML-Format vor. Dieses muss nun von der Erweiterung „SimpleXML“ eingelesen werden. Dabei wird die XML-Struktur in ein PHP-Objekt umgewandelt, auf das nun wie ein Array zugegriffen werden kann.

3. children()

    - Der dritte Schritt braucht die Funktion *children()*. Sie wird benötigt, falls die XML-Elemente über Namensräume, sog. „namespaces“, enthalten. Der Funktion wird die Variable *$metadata* übergeben. Sie enthält die Adresse, wo die in der XML-Datei verwendeten Namensraumdefinitionen zu finden sind.

Wurden diese drei Stufen durchlaufen, kann das Ergebnis nun wie erwähnt verarbeitet und formatiert ausgegeben werden. Die Ausgabe hat folgendes Format:

Autor(, weitere(r) Autor(en)) (Jahr der Veröffentlichung in Pangaea): Titel
Beispiel: *König-Langlo, G (2005): Meteorological observations during POLARSTERN cruise ANT-XI/3*

Traten hingegen bei dem Web Service-Aufruf Fehler auf, werden anstelle der Ergebnisliste Fehlermeldungen gezeigt. Eine andere Möglichkeit ist, dass keine Ergebnisse vorliegen, weil keine gefunden wurden. Hierüber wird dem Benutzer ebenso eine entsprechende Meldung angezeigt.

5.3.7.2 Verarbeitung der Suchergebnisse von Fedora

Für die Anzeige der Fedoradaten ist die Funktion *fedora_res()* zuständig. Ihr werden die Werte *$result_fedora* (das vom Web Service gelieferte Ergebnis) und *$fedora_error* (Fehlerstatus) übergeben. Wie schon bei Pangaea werden die Ergebnisse per Schleife durchlaufen und ausgegeben. Das Objekt, welches der Web

[9] Base64 beschreibt ein Verfahren zur Kodierung von 8-Bit-Binärdaten in eine Zeichenfolge, welche aus nur wenigen ASCII-Zeichen besteht.

Service liefert, muss aber im Gegensatz zu dem von Fedora nicht mehr bearbeitet, sondern kann direkt benutzt werden.

Das Ausgabeformat ist aus Gründen der Einheitlichkeit das Gleiche wie bei der Ausgabe der Datensätze. Ebenso werden Meldungen ausgegeben, falls ein Fehler aufgetreten ist oder die Suche keine Ergebnisse erzielt hat.

5.3.7.3 Verarbeitung der Details von Pangaea

Der Aufruf der Details eines in Pangaea gespeicherten Eintrags erfolgt wieder über eine Methode im Web Service. Um diese Methode aufzurufen wird der Funktion *show_detail()* in der Klasse *webservice_pangaea* der DOI übergeben. In dieser Funktion wird dann die Web Service-Methode *metadata()* mit der Pangaea-Session und besagter DOI aufgerufen und das Ergebnis zurückgeliefert.

Dieses Ergebnis muss ebenfalls wieder dekodiert, als XML-String eingelesen und mit der Definition der Namensräume ausgestattet werden. Erst dann kann eine weitere Arbeit damit erfolgen. Für die Ausgabe werden nun die einzelnen Bestandteile des Ergebnisses ausgelesen und formatiert. Neben den Informationen zu den Datensätzen wird noch eine Tabelle erstellt, welche sämtliche Parameter der gemessenen Daten enthält.

Eine detaillierte Anfrage für ein Ergebnis liefert in den meisten Fällen sehr viele Einträge zurück, die alle abgefangen werden müssen. Gemäß dem Pflichtenheft muss also verhindert werden, dass Details bei der Anzeige „verloren gehen“.

Die Elemente befinden sich aufgelistet in einer von Pangaea zur Verfügung gestellten XSD-Datei[10] (XML Schema Definition). Die Abbildung 5.6 zeigt beispielhaft die Struktur des Elementes *citation*. Eine Auflistung aller Elemente an dieser Stelle wäre zu umfangreich, die vollständige XSD-Datei befindet sich im Anhang auf Seite 189.

[10] http://ws.pangaea.de/xml/MetaData.xsd

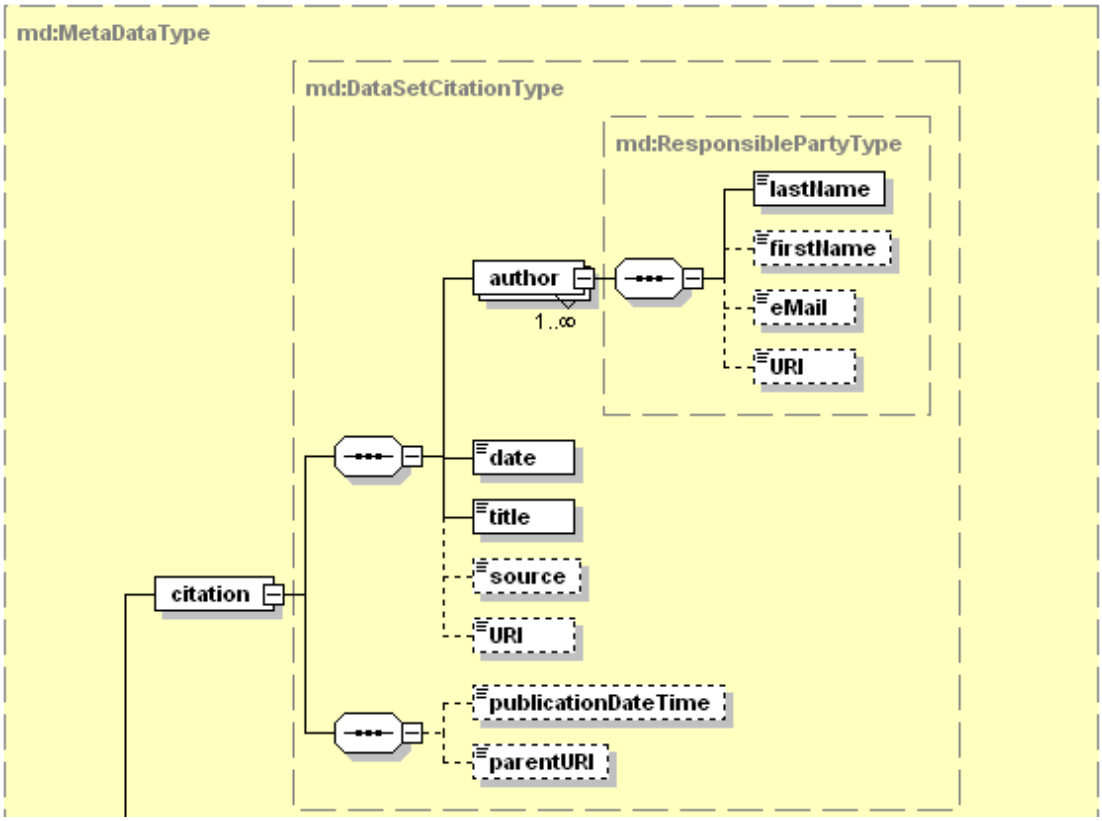


Abbildung 5.6: Portal: Struktur des Elementes *citation*, vgl. [Pan05 b]

Wie die Abbildung 5.6 zeigt, enthält das Element *citation* die Unterelemente *author* (Autor), *date* (Datum), *title* (Titel) – alles obligatorische Angaben – sowie *source* (Quelle(n)), *URI* (DOI), *publicationDateTime* (Datum der Veröffentlichung) und *parentURI* (übergeordnete URI) – optionale Angaben. Das Unterelement *author* teilt sich dann noch einmal in *lastName* (Nachname), *firstName* (Vorname), *eMail* (Emailadresse) und *URI* (Homepage des Autors).

Besonderes Augenmerk sei auf die Funktion *dataset()* gerichtet. Sie stellt dem Benutzer die Möglichkeit zur Verfügung, sich die Daten anzeigen zu lassen und herunterzuladen. Vorher werden der Funktion die Werte *$show_tab* (Messdaten anzeigen ein/aus), *$doi* (der DOI), *$xml_str→size* (Anzahl der Datenpunkte) und *$login[0]* (Anmeldung erforderlich ja/nein) übergeben. Zu *$login[0]* sei noch gesagt, dass dieser Wert über XPath[11] ermittelt wurde, einer vom W3C entwickelten Anfragesprache (und SimpleXML-Bestandteil) zum direkten Adressieren von Teilen eines XML-Dokumentes. Auf diesen Teil des Pangaea-Ergebnisses konnte mit der einfachen SimpleXML-Behandlung nicht zugegriffen werden.

[11] XML Path Language

Die Funktion erzeugt nun zwei Verweise und einen Button. Beim Aktivieren des Buttons wird die Seite neu geladen, diesmal mit einer zusätzlichen Tabelle, welche die Messdaten enthält. Da diese Tabelle sehr groß werden kann, wird der Benutzer vorher drauf aufmerksam gemacht, dass längere Ladezeiten möglich sind (abhängig von der Internetanbindung des Benutzers).

Über diesen Button kann die Anzeige auch wieder ausgeschalten werden. Hinter den beiden Verweise stehen zum Einen die zu den Daten gehörige CSV-Datei, gehostet von Pangaea selbst, zum Anderen das Skript *xmlbuilder.php* zur dynamischen Erzeugung einer XML-Datei. Dieses Skript liest die Messdaten in der CSV-Datei ein, wandelt sie in XML um und bietet sie dem Benutzer zum Herunterladen an. Ein Beispiel für eine erzeugte XML-Datei ist in Quelltext 5.11 zu sehen. Um ein in der Wissenschaft gebrauchtes Format zu verwenden, wurde die XML-Generierung so weit wie möglich an das MarineXML-Schema angelehnt. MarineXML wurde vom australischen ozeanographischen Datenzentrum (AODC[12]) für die Verwaltung maritimer, wissenschaftlicher Daten entwickelt. Dieses spezielle XML-Format eignet sich aber auch für andere Daten, wie hier z.B. für die Meteorologie. Das komplette Schema des Formats bzw. dessen Beschreibung kann im Internet abgerufen werden[13].

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE MarineDataSet SYSTEM "http://web.awi-bremerhaven.de//php/MISAWIsta/
       MarineXML/MarineXML_Ver2.0.dtd">
3  <MarineDataSet creationDate="2005-09-01T15:38:36+02:00" name="doi:10.1594/PANGAEA
       .57692" description="Silt fraction analysis of sediment core PS1385-3">
4    <Quality>Good</Quality>
5    <Custodian>
6      <Property name="Agency">Pangaea Information System</Property>
7      <Property name="WebSite">http://www.pangaea.de</Property>
8    </Custodian>
9    <MarineDataRecord ID="0" reject="false">
10     <SpatialReference>
11       <GeoBox>
12         <Coordinates datum="WGS84">
13           <Latitude>-70.500</Latitude>
14           <Longitude>-9.617</Longitude>
15         </Coordinates>
16         <Coordinates datum="WGS84">
17           <Latitude>-70.500</Latitude>
18           <Longitude>-9.617</Longitude>
19         </Coordinates>
20       </GeoBox>
21     </SpatialReference>
22     <Source isObservedDate="true" sourceFileName="http://www.pangaea.de/ddi?
           datasetid=57692&amp;format=textfile" agency="Pangaea" projectID="57692">
23     </Source>
24     <Data numberOfDataObjects="51">
25       <DataObject index="0" type="Primary" reject="false">
26         <ParameterSet index="0" numberOfParameters="0">
27           <ValueList numberOfValueSets="12">, Depth [m], 0.01, 0.10, 0.20, 0.30,
                 0.40, 0.50, 0.60, 0.70, 0.80, 0.85</ValueList>
28         </ParameterSet>
```

[12] vgl. http://www.aodc.gov.au/AODC.html
[13] vgl. http://www.metoc.gov.au/products/prod/documentation/marine_xml_schema.html

```
29        </DataObject>
30        ...
31        <DataObject index="50" type="Primary" reject="false">
32          <ParameterSet index="0" numberOfParameters="0">
33            <ValueList numberOfValueSets="12">PS1385-3.94, 8.9-9.0phi [%], 0.14,
                  0.17, 0.16, 0.39, 0.16, 0.17, 0.16, 0.28, 0.25, 0.29</ValueList>
34          </ParameterSet>
35        </DataObject>
36      </Data>
37    </MarineDataRecord>
38  </MarineDataSet>
```

Quelltext 5.11: Portal: Dynamisch erzeugte XML-Datei (gekürzter Auszug)

5.3.7.4 Verarbeitung der Details von Fedora

Die Verarbeitung von Fedora-Daten ist ähnlich aufgebaut wie die von Pangaea. Nur das hier wieder auf die Dekodierung, XML-Verarbeitung und die Adressierung der Namensräume verzichtet werden kann. Bei der Formatierung ist zu beachten, das beim Auftreten mehrerer Einträge (z.B. mehrere Autoren) der Zugriff auf die Daten über eine Schleife erfolgen muss, um alle Daten aus dem entsprechenden Array zu erreichen. Bei nur einem Eintrag ist dies nicht notwendig.

Nicht im Ergebnis enthalten ist der sog. „Abstract“, die kurze Zusammenfassung der Publikation. Dieser wird über eine regulären Ausdruck auf der Seite, auf dem die Beschreibung steht, eingelesen. Da sich der gesamte „Abstract“ zwischen zwei HTML-Kommentar-Tags befindet, lässt sich dieser wie in Quelltext 5.12 gezeigt einlesen.

```
1 if ( !is_array($item->description) && $item->description != null )
2 {
3   $ext_abstract = file_get_contents(ltrim($item->description, "uri:"), "r");
4   preg_match_all("/<!--begin_of_abstract-->(.*?)<!--end_of_abstract-->/ims",
        $ext_abstract, $abstract);
5 }
```

Quelltext 5.12: Portal: Einlesen des Abstract

5.3.8 Gestaltung

Die Gestaltung des Portals erfolgte über eine separate CSS-Datei. Genauer gesagt aus drei, denn die Gestaltung des *head* und des *body*-HTML-Elements ist aufgespaltet in eine für den Internet Explorer und eine für die anderen Browser. Hier macht sich wieder das nichtkonforme Verhältnis des IE zu gängigen W3C-Standards bemerkbar. Es wird beim Aufbau der Seite überprüft, ob es sich bei dem verwendeten Browser um den IE oder einen anderen Typ handelt, dementsprechend wird Kopfdatei geladen. Der Hauptteil aber ist für alle Browser gleich,

es wurde darauf geachtet, dass die benutzten Elemente von allen Browsern interpretiert werden können. Die Entwicklung der CSS-Datei erfolgte direkt neben der Entwicklung der Ausgabebestandteile des Portals. Deshalb erfolgten ständig Änderungen und Anpassungen an die Gegebenheiten des jeweiligen Skripts. Was am Ende der Gestaltungsarbeiten herauskommen ist, kann man am Portals selbst am Besten sehen.

Der Footer des Portals enthält vier Graphiken, die verschiedene Funktionen vorweisen. Mit dem ersten Button (W3C XHTML 1.1) kann überprüft werden, ob es sich bei der angezeigten Seite um valides, also gültiges XHTML 1.1 handelt. Der zweite Button (W3C CSS 2) überprüft die syntaktische Korrektheit der verwendeten CSS-Datei. Die erfolgreichen Testergebnisse sind auf den Abbildungen 5.7 und 5.8 zu sehen. Zu vermerken ist noch, dass die Validierung von XHTML bei jeder Seite einzeln vorgenommen werden muss. Von daher wurde hier nur die Startseite zu exemplarischen Zwecken verwendet. Dann folgt der Button (W3C WAI AA) für die WAI-Kompatibilität. Zum Schluss der Button RSS NEWSFEED, hinter welchem sich der RSS-Feed (s. S. 85) bzw. die dafür verwendete XML-Datei verbirgt.

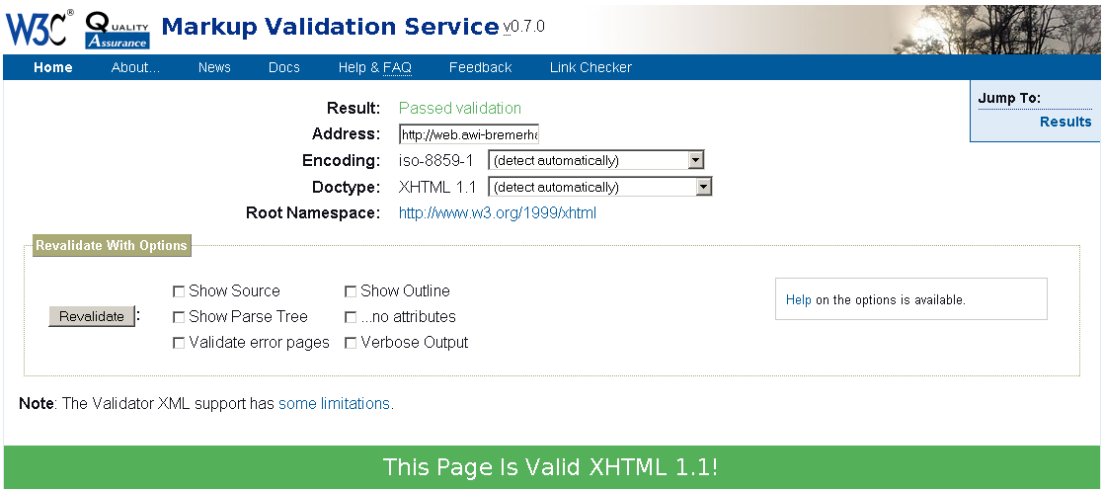


Abbildung 5.7: Portal: Validiertes XHTML 1.1

Abbildung 5.8: Portal: Validiertes CSS

5.3.9 Abschließende Entwicklungen

Nachdem die grundlegende Arbeit mit den Web Services abgeschlossen ist, wird das Portal noch um einige Bestandteile erweitert. Schließlich soll es ja für den Benutzer auch einfach zu handhaben sein.

5.3.9.1 Navigator

Die erste Entwicklung für ein komfortablen Umgang ist der Navigator. Dieser ermöglicht das vor- und zurückblättern zwischen den Seiten mit den Ergebnislisten. Da die Variable *$count* auf zehn Ergebnisse pro Liste festgesetzt ist, oft aber weit mehr diese bei einer Anfrage zurückgeliefert werden, wird diese Funktion dringend benötigt. Es ist ja nicht vertretbar, dem Benutzer nur eine Seite mit 20 Ergebnissen anzubieten, wenn in Wirklichkeit z.B. mehr als 100 existieren. Die Abbildung 5.9 zeigt den vollständigen Navigator.

Abbildung 5.9: Portal: Navigator

Hinter dem Navigator stehen mathematische Berechnungen. Die Funktion erhält die Werte der Variablen *$total_count* (größtmögliche Anzahl der Ergebnisse), *$offset* (Beginn des Bereichs der Ergebnisauflistung) und *$count* (Ergebnisse pro Liste) als Parameter. Aus diesen wird nun der Navigator „errechnet“ (vgl. Quelltext 5.13).

Ist der Wert von *$total_count* kleiner als der von *$count*, wird kein Navigator ausgegeben. Ist er allerdings größer, wird der Navigator angezeigt. Dabei wird zwischen *NEXT* (vorwärts) und *PREV* (zurück) unterschieden. Der *NEXT*-Link erscheint nur solange auf der nachfolgenden Seite Ergebnisse existieren (alle außer der letzten), der *PREV*-Link dagegen nur, wenn auf der vorhergehenden Seite Einträge zu finden sind (alle außer der ersten).

```
1  # turn over the pages
2  function navigator( $total_count, $offset, $count )
3  {
4    # calculate some variables
5    $maxpages = $total_count/$count;
6    $page=$offset/$count;
7
8    if ( $total_count > $count )
9    {
10     # produces the PREV-button
11     echo "<div class=\"navigator\">\n";
12     if ( $page > 0 )
13     {
14       $prev = ($page-1)*$count;
15       if ($prev <= 0) $prev = 0;
16
```

```
17        echo "   <a href=\"search.php?offset=".$prev."\">"
18             ."     &lt;&lt; PREV"
19             ."   </a>\n";
20      }
21
22      echo "<span style=\"color:#006ba5\"> | </span>";
23
24      # produces the NEXT-button
25      if ( $page < $maxpages - 1 )
26      {
27        $next = ( $page+1 )*$count;
28
29        echo "   <a href=\"search.php?offset=".$next."\">"
30             ."     NEXT &gt;&gt;"
31             ."   </a>\n";
32      }
33      echo "</div>\n";
34    }
35  }
```

Quelltext 5.13: Portal: Navigator

Bei Benutzen der *PREV/NEXT*-Verweise wird der Wert der Variable *$offset* um den von *$count* verringert oder erhöht. Für die Voreinstellung würde das bedeuten -10 oder +10.

5.3.9.2 Sidebar

Die Sidebar ist das zweite Steuerungselement. In erster Linie dient sie dem Nutzer als Informationsquelle. Es werden der neben der gesuchten Phrase auch die Anzahl der gefundenen Ergebnisse untergebracht (s. Abb. 5.10).

Abbildung 5.10: Portal: Sidebar

Neben passiven Funktionen kann der Benutzer hier auch die Anzeige der Ergebnisse einschränken. In den meisten Fällen ist „list all“ (die Anzeige aller Ergebnisse) voreingestellt. Möchte der Benutzer sich aber nur die Datensätze („list datasets“) oder nur die Publikationen („list publications“) anschauen, genügt der Klick auf den entsprechenden Link.

Wird ein solcher Link aufgerufen, werden die Variablen *$pub* oder *$data* verändert (0→1 bzw. 1→0), s. Quelltext 5.14. Damit wird beim Generieren der jeweiligen Ergebnisliste nur die Ausgewählte angezeigt. Um zu gewährleisten, dass der Benutzer bei einer erneuten Änderung die korrekten Ergebnislisten wieder angezeigt bekommt, werden alle Aktionen wie bei der Anzeige aller Daten behandelt, sie laufen nur im Hintergrund ab.

```
1  # save the links in variables for a better handling
2  $d1p0 = "<a href=\"search.php?data=1&amp;pub=0&amp;offset=".$offset."\">list
       datasets</a><br />";
3  $d0p1 = "<a href=\"search.php?data=0&amp;pub=1&amp;offset=".$offset."\">list
       publications</a><br />";
4  $d1p1 = "<a href=\"search.php?data=1&amp;pub=1&amp;offset=".$offset."\">list all
       </a>";
5
6  echo "<div>";
7  if ( $_SESSION["data"] == 1 && $_SESSION["pub"] == 1 )
8  {
9    echo $d1p0;
10   echo $d0p1;
11   echo "<b>list all</b><br />";
12 }
13 else if ( $_SESSION["data"] == 0 && $_SESSION["pub"] == 1 )
14 {
15   echo $d1p0;
16   echo "<b>list publications</b><br />";
17   echo $d1p1;
18 }
19 else if ( $_SESSION["data"] == 1 && $_SESSION["pub"] == 0 )
20 {
21   echo "<b>list datasets</b><br />";
22   echo $d0p1;
23   echo $d1p1;
24 }
25 # if data and pub-variable is 0 get back to search_advanced.php
26 # this can happen if both checkboxes in a advanced search were deselected
27 else
28 {
29   header("Location: search_advanced.php?query=".$_SESSION["query"]."");
30   exit;
31 }
```

Quelltext 5.14: Portal: Sidebar (Auszug)

Die Sidebar fängt auch den Fall ab, dass ein Benutzer (versehentlich) bei der erweiterten Suche *beide* Checkboxen („list datasets“, „list publications“) abgewählt hat. In dem Fall wird sofort wieder die Datei *search_advanced.php* aufgerufen.

5.3.9.3 Nachrichtenseite / RSS-Feed

Damit die Benutzer über Änderungen informiert werden können (z.B. über neue Funktionen von MISAWIsta oder Wartungsarbeiten bei den verwendeten Web Services) ist eine Informationsseite angebracht. Diese ist über den Punkt *News* in der Fußzeile der Seite zu finden. Hierbei wurde aber nicht auf festen Inhalt,

sondern auf die RSS-Technologie gesetzt. RSS ist eine auf XML basierende Familie von Dateiformaten. Die Abkürzung steht, je nach Version, für:

- Rich Site Summary / RSS 0.9x
- RDF Site Summary / RSS 1.0
- Really Simple Syndication / RSS 2.0

Dabei werden Nachrichten in maschinenlesbarer Form (XML) gespeichert. Die Nachrichten enthalten wie bei XML üblich keine Formatierung, diese bleibt den Webautoren überlassen. Die XML-Dateien können von speziellen Programmen („Feedreadern“) eingelesen und dargestellt werden. Neben diesen Anwendungen bieten auch einige Email-Applikationen (wie Mozilla Thunderbird) die Unterstützung von den sog. „RSS-Feeds“. Für MISAWIsta werden drei Schritte benötigt, um einen RSS-Feed zur Verfügung zu stellen.

Erstellung der XML-Datei Die XML-Datei besteht aus zwei Teilen. Der Header, welcher Meta-Informationen über die gesamte Seite enthält, z.B. URL oder die verwendete Sprache, und der Body, welcher die Nachrichten in *item*-Elementen enthält (vgl. Quelltext 5.15). Den *item*-Elementen wurde das Attribut *date* hinzugefügt, welches das Datum der Nachricht kennzeichnet. Dies wird benötigt für den zweiten Schritt.

```
1  ?xml version="1.0" encoding="ISO-8859-1" ?>
2  <rss version="2.0">
3    <channel>
4      <title>MISAWIsta News</title>
5      <link>http://web.awi-bremerhaven.de/php/MetVista/news.php</link>
6      <description>Newspage of MISAWIsta</description>
7      <language>en-en</language>
8      <copyright>2005 AWI Bremerhaven</copyright>
9      <pubDate>2005-08-19</pubDate>
10     <image>
11       <url>http://web.awi-bremerhaven.de/php/MetVista/pics/banner.jpg</url>
12       <title>MISAWI Banner</title>
13       <link>http://www.awi-bremerhaven.de</link>
14     </image>
15     <item newsdate="2005-08-19">
16       <title>RSS-feed available</title>
17       <description>Today, a RSS-feed for the news was implemented. Now it is
             possible to get the latest news for MISAWIsta with an RSS-Reader (or
             applications which support RSS, like &lt;a href="http://www.mozilla.org
             /products/thunderbird/"&gt;Mozilla Thunderbird&lt;/a&gt;).</description
             >
18       <link>http://web.awi-bremerhaven.de/php/MetVista/news.php?date=2005-08-19</
             link>
19     </item>
20   </channel>
21 </rss>
```

Quelltext 5.15: Portal: RSS-Feed (Auszug)

Erstellung der Nachrichtenseite Die Nachrichtenseite kann auf zweierlei Arten verwendet werden. Die erste Art ist der normale Aufruf über den Link der Webseite. Dabei werden alle in der XML-Datei gespeicherten Nachrichten aufgerufen und dargestellt. Die zweite Art ist die Anzeige über einen Feedreader bzw. ein RSS-fähiges Programm. Dabei wird die Nachricht ausgewählt und aufgerufen. Durch die Übergabe des Datums wird nun nur jene Nachricht angezeigt, deren Attribut *date* mit dem übergebenen Datum übereinstimmt.

In beiden Fällen liest das Skript *news.php* die XML-Datei ein (erneut über SimpleXML) und verarbeitet die Informationen. Je nachdem, ob ein Datum übergeben wurde oder nicht, gibt das Skript alle oder die gewählte Nachricht aus.

Eintrag in Header Nicht notwendig, aber sinnvoll ist das Setzen eines Verweises auf den RSS-Feed in den Header der Seite. Moderne Browser können RSS verarbeiten und sog. *dynamische Lesezeichen* auf sie setzen. Damit können die einzelnen(!) Nachrichten auch über den Browser betrachtet werden. Um diese zu erreichen, muss die in Quelltext 5.16 gezeigte Zeile 1 in das *head*-Element der Seite eingetragen werden. (Die zweite Zeile ist der RSS-Feed von Pangaea selbst. Er wurde aufgenommen, um das Angebot von MISAWIsta zu erweitern.)

```
1 <link rel="alternate" title="Latest MISAWIsta News" href="http://web.awi-
      bremerhaven.de/php/MetVista/feed/MISAWIsta-news.rss" type="application/rss+
      xml" />
2 <link rel="alternate" title="Latest Pangaea Datasets" href="http://www.pangaea.de
      /PangaVista/latest-datasets.rss" type="application/rss+xml" />
```

Quelltext 5.16: Portal: RSS-Eintrag im Header

Um die jeweils neueste Information dem Benutzer zugänglich zu machen, wurde auf der Startseite eine entsprechende Box eingebaut. Somit wird dem Nutzer z.B. direkt mitgeteilt, ob zur Zeit eine Suche überhaupt möglich, weil vielleicht gerade ein System gewartet wird et cetera.

5.3.9.4 Informationsseiten

Zum Schluss wurde neben dem Impressum, der Nachrichtenseite und den Verweisen eine Seite geschrieben, welche die Hilfe für das System enthält. Der Benutzer bekommt hier eine Hilfestellung zur Systembenutzung und Antworten auf evt. Fragen (FAQ). Der Quelltext dafür befindet sich im Anhang auf Seite 139.

5.4 Funktionstest

Ein wichtiger Bestandteil der Softwareentwicklung ist die Verifikation des Programms, d.h. der Beweis seiner Korrektheit. Die Testverfahren haben das Ziel,

Fehler zu erkennen. Diese Verfahren lassen sich aufteilen in dynamische (Strukturtestverfahren (White Box-Test, Glass Box-Test), funktionale Testverfahren (Black Box-Test)) und statische Testverfahren (Walkthrough, Inspektion). Bei den dynamischen Verfahren werden dem Programm konkrete Eingabewerte übergeben. Es wird dann in einer „realen“ Umgebung getestet. Bei statischen Testverfahren hingegen wird das Programm nicht ausgeführt, sondern der Quellcode analysiert, um Fehler zu finden.[14]

Bei der Entwicklung (Implementierung) des Portals wurde bereits darauf geachtet, die Ergebnisse von Funktionen auf ihre Korrektheit hin zu überprüfen. Das heißt, es wurde schon mit der White Box-Methode getestet. Der White Box-Test ist geeignet, um Fehler in den Komponenten aufzudecken und zu lokalisieren. Black Box-Tests, sprich Tests ohne Blick auf die innere Funktionsweise zur Überprüfung der Spezifikationseinhaltung, würden hier keinen Nutzen bringen.

Die Tests, die zeitgleich mit der Funktionsentwicklung stattfanden, verliefen nach einem bestimmten Schema. Die Funktion, unabhängig davon ob allein stehend oder in einer Klasse befindlich, wurde ggf. mit Übergabe von Parametern aufgerufen. Danach ist die Ausgabe bzw. der Rückgabewert der Funktion kontrolliert worden. Zu diesem Zweck wurde eine sog. Debug-Funktion in das Programm mit aufgenommen. Diese Funktion gibt nun einen formatierten Speicherauszug (dump) der übergebenen Variable aus. Damit wird überprüft, ob die Variable das gewünschte Ergebnis enthält. Falls nicht, ist der Quelltext zu korrigieren.

Als weitere Testart wurde das statische Verfahren der Quelltextanalyse eingesetzt. Dabei wurde im Quelltext überprüft, ob Funktionen, welche mit anderen Funktionen interagieren, dies auch korrekt durchführen. Der praktische Test erfolgte dann wieder nach der White Box-Methode. Da es sich bei dem Portal um eine Ansammlung von dynamischen Webseiten handelt, war auch das „optische Verfahren“ (durchlesen, überprüfen und ggf. berichtigen) eines der Hauptmittel beim Testen der Funktionen. Es musste überprüft werden, ob die Ergebnisse korrekt formatiert dargestellt sind, dass keine Informationen unterschlagen wurden et cetera. Dafür blieb nur die direkte Gegenüberstellung des Speicherauszuges der Variable und der daraus erstellen Ausgabe, um einen Fehler zu bemerken bzw. zu analysieren.

5.5 Inbetriebnahme und Demonstration

Nach erfolgreichem Abschluss der Funktionstests erfolgt die Inbetriebnahme der Software. Diese ist zwar durch die Arbeit auf dem Produktivsystem bereits weit früher erfolgt, aber das V-Modell sieht diesen Punkt am Ende der Arbeiten vor.

[14] vgl. [Balzert 1999b, S. 509f]

5.5.1 Ablauf

Vor der Demonstration wird der Programmablauf während der Nutzung des Portals als EPK dargestellt. Die Verwendung der Seiten „Links“, „About“ und „Help“ wird nicht in das EPK mit aufgenommen, da sie nur statische Informationen ohne für den Ablauf wichtige Funktionen bieten. Die Verwendung der Nachrichtenseite „News“ bzw. von RSS ist in Abbildung 5.14 auf Seite 92 gesondert wiedergegeben.

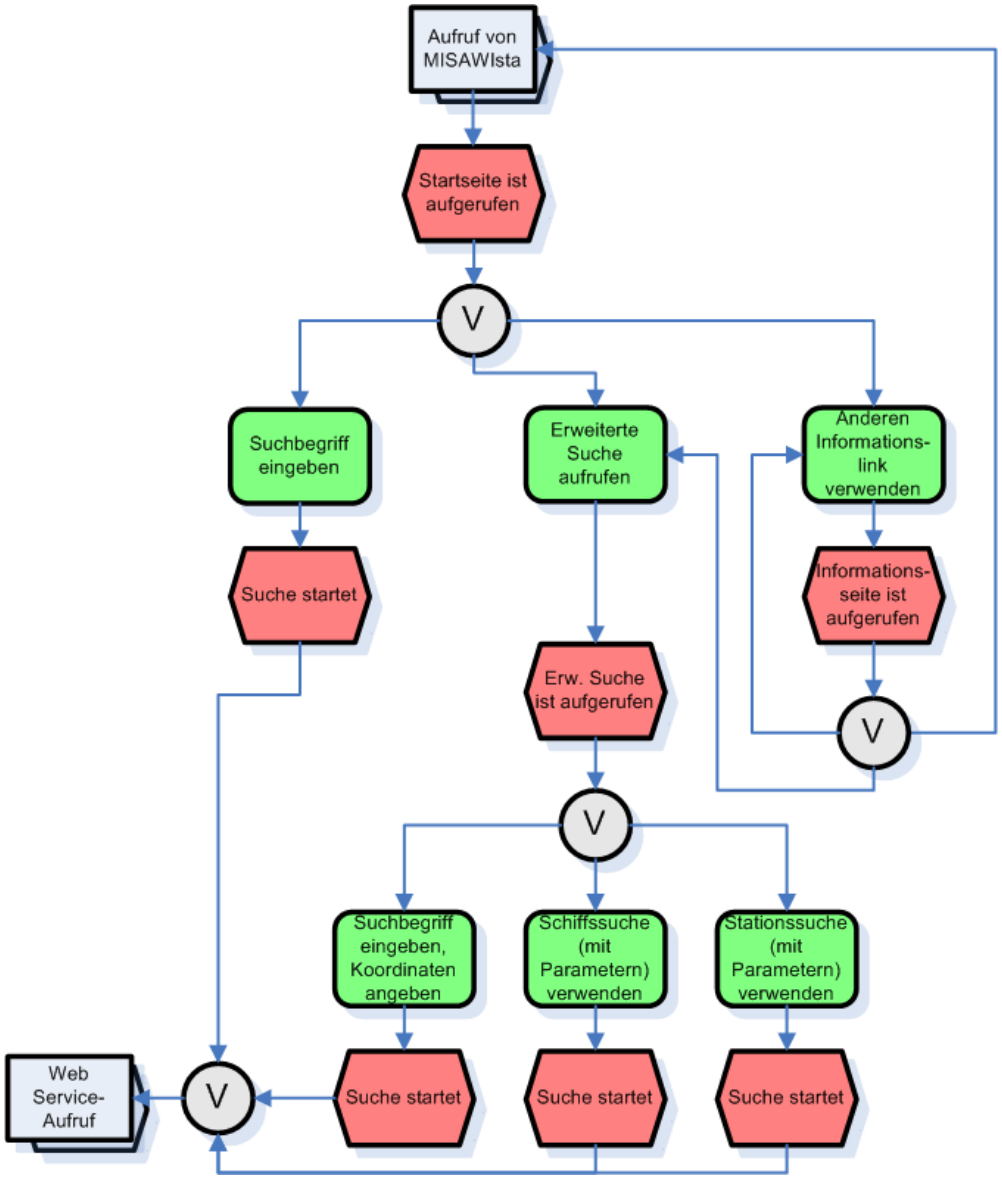


Abbildung 5.11: EPK des Portals, Teil 1

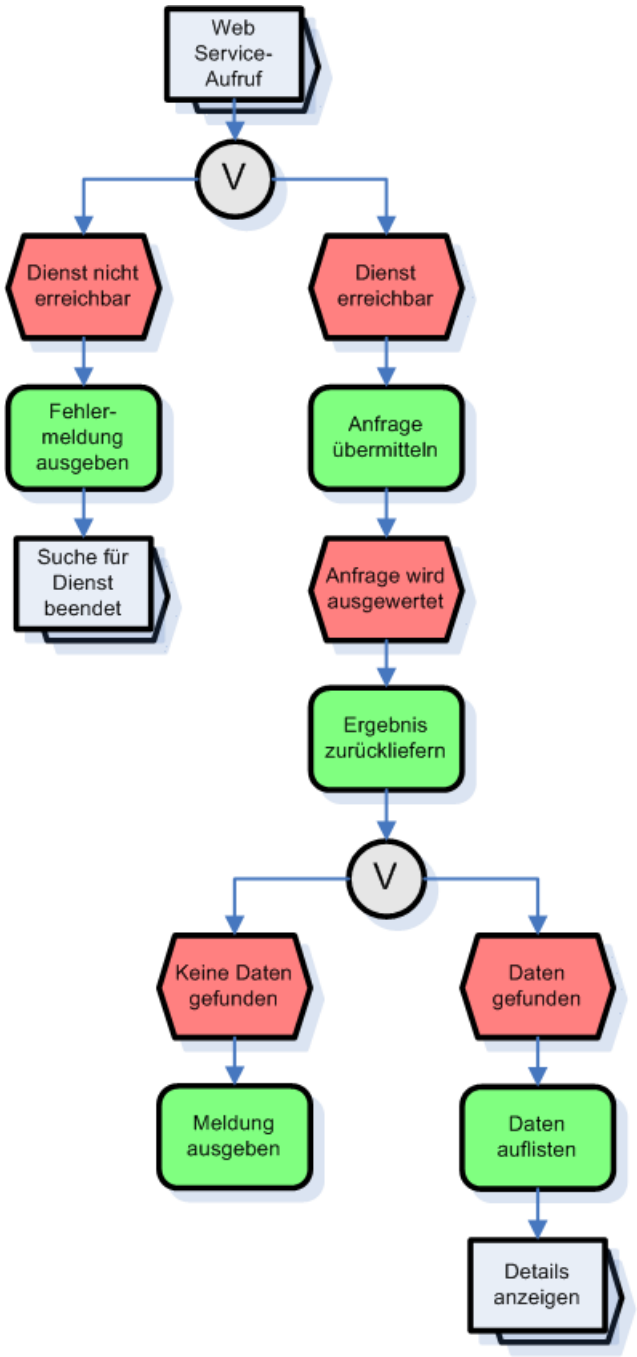


Abbildung 5.12: EPK des Portals, Teil 2

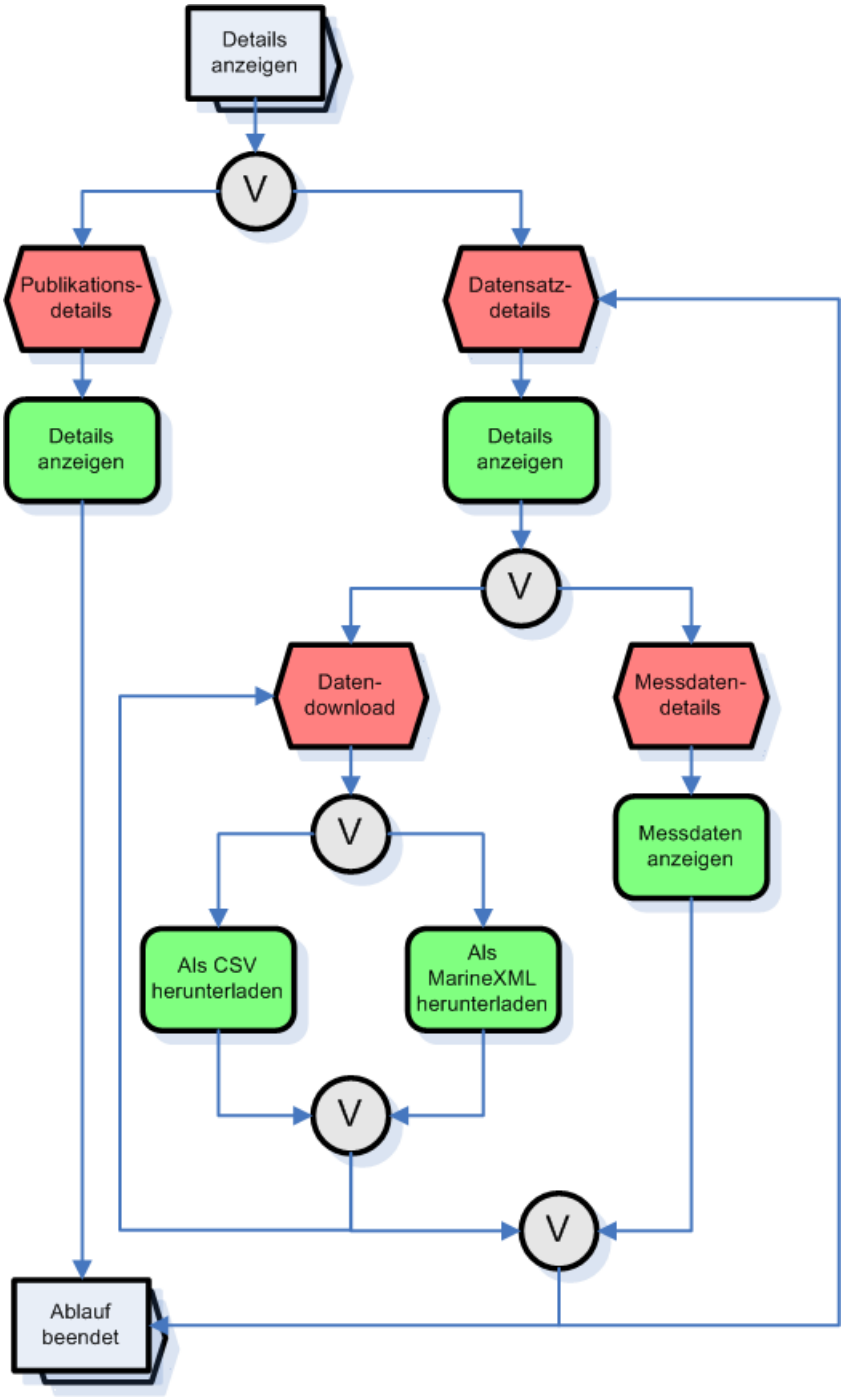


Abbildung 5.13: EPK des Portals, Teil 3

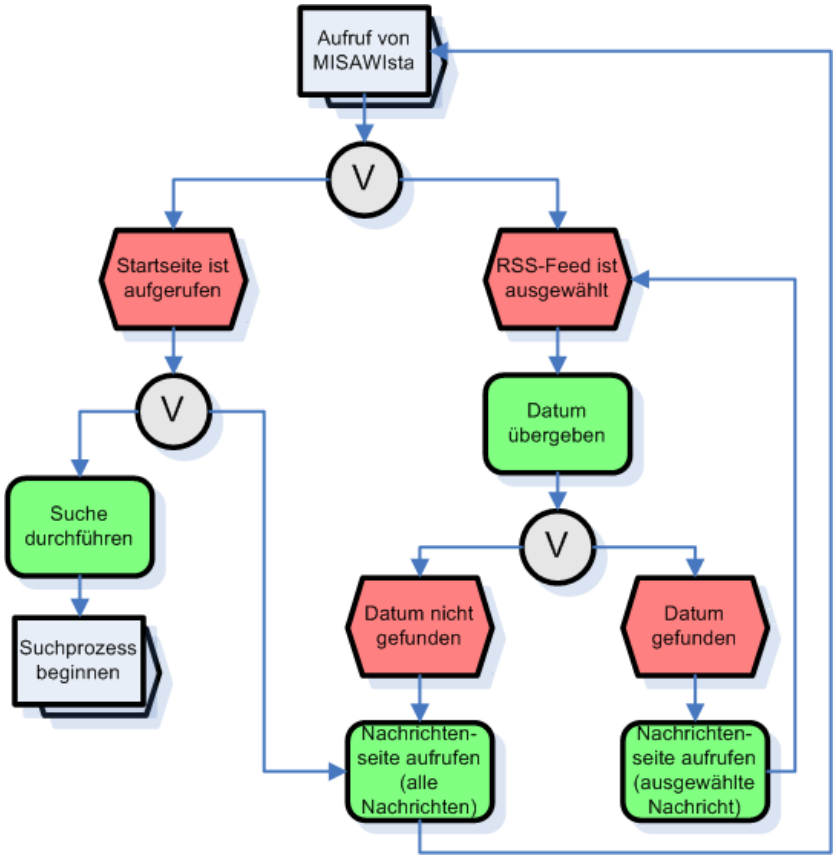


Abbildung 5.14: EPK des RSS-Feed

5.5.2 Demonstration

Es folgt eine Demonstration des Systems mit Abbildungen. Dabei werden verschiedene Szenarien durchgespielt, die bei der Nutzung eintreten können.

5.5.2.1 Szenario 1: Einfache Suche mit Detailansicht

Das erste Beispiel ist die einfache Suche eines Benutzers nach Informationen zu der Neumayer-Station (Suchbegriff „neumayer“). Der erste Schritt ist der Aufruf der Startseite (s. Abb. 5.15).

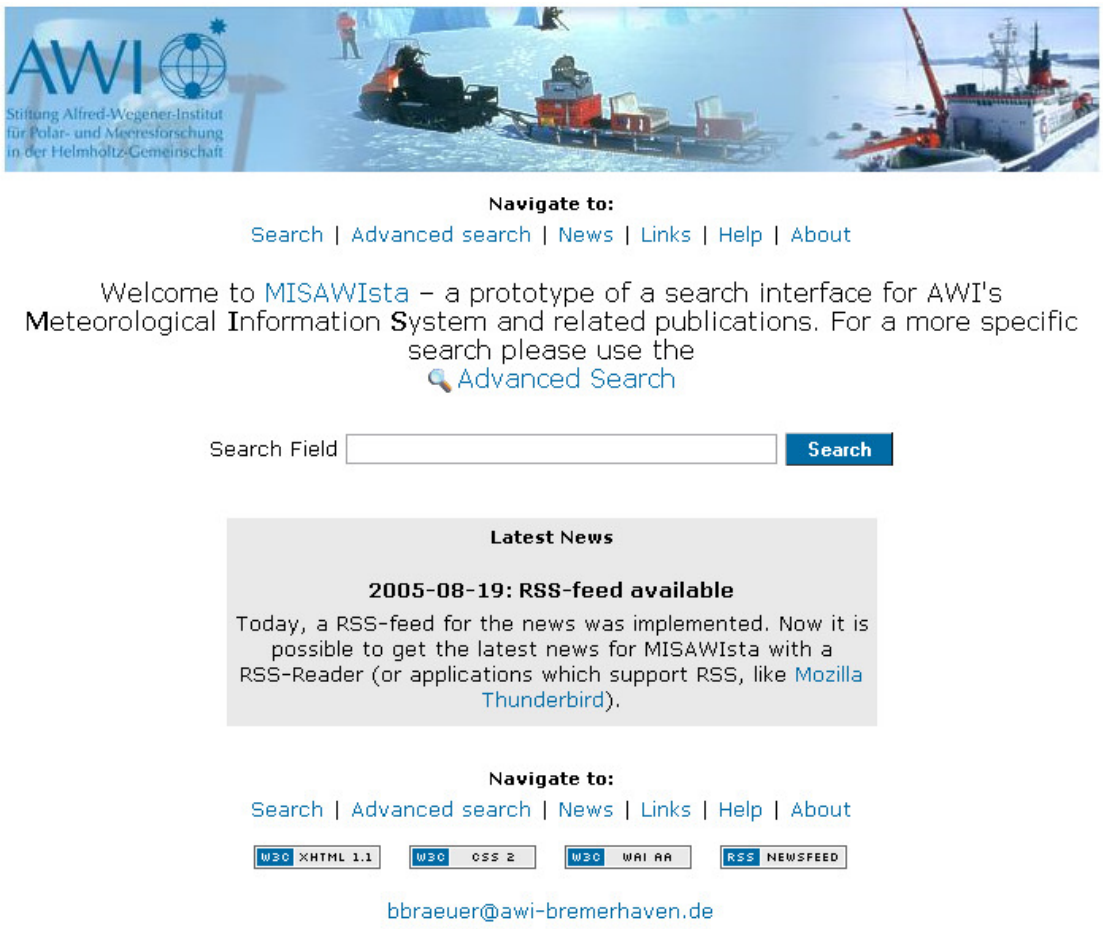


Abbildung 5.15: Demonstration: 1 – Aufruf der Startseite / Eingabe Suchbegriff

Der Benutzer gibt den Suchbegriff in das dafür vorgesehene Feld ein. Ein Klick auf den Button „Search“ startet den Suchvorgang.

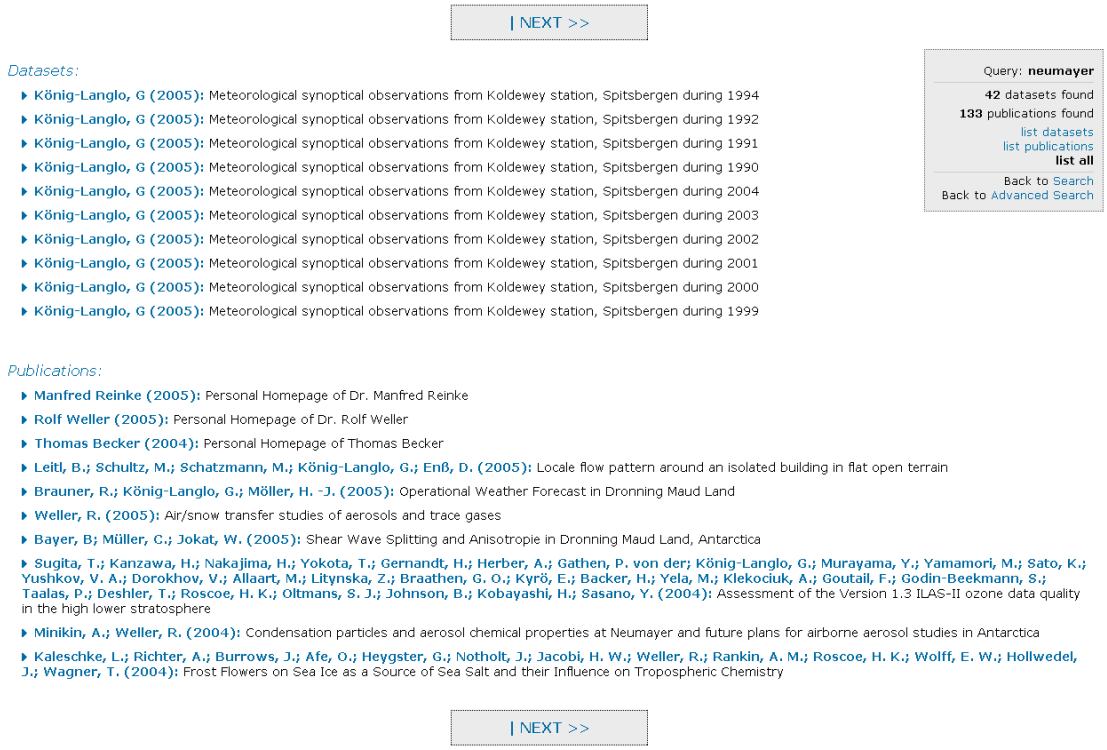


Abbildung 5.16: Demonstration: 1 – Ergebnislisten der Suche „neumayer“

Ist die Suche erfolgreich abgeschlossen, werden dem Benutzer die Ergebnislisten präsentiert. Diese sind für das verwendete Beispiel in Abb. 5.16 gezeigt.

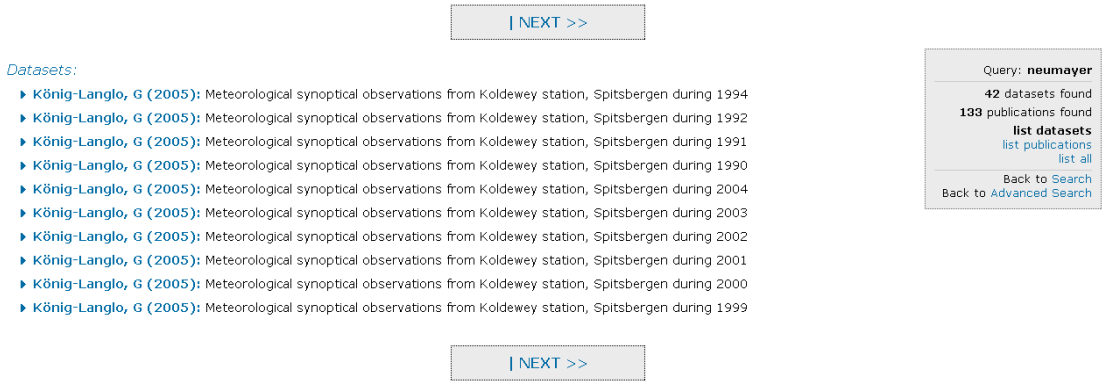


Abbildung 5.17: Demonstration: 1 – Ergebnisliste Dataset

Über die Sidebar lässt sich auch die Anzeige der Listen steuern. In Abb. 5.17 werden nur Datensätze abgebildet. . .

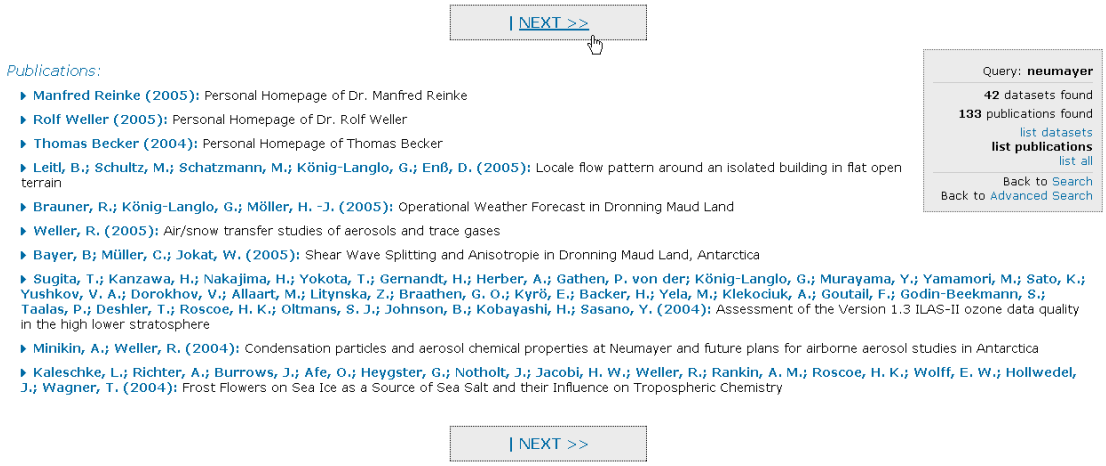


Abbildung 5.18: Demonstration: 1 – Ergebnisliste Publication, Seite 1

. . . wogegen in Abb. 5.18 nur die Publikationen zu sehen sind. Nun soll noch kurz der Navigator gezeigt werden, auf dem der Mauszeiger gerade deutet. Wird der *NEXT*-Link gedrückt, werden die nächsten zehn Publikationen gezeigt, zu sehen in Abb. 5.19.

Abbildung 5.19: Demonstration: 1 – Ergebnisliste Publication, Seite 2

Der Navigator hat sich der Seitenänderung angepasst und bietet nun auch die Möglichkeit, per *PREV* zurückzublättern. Inzwischen hat der Benutzer seine gesuchten Daten gefunden und möchte diese im Detail betrachten.

Record Details

Title:	Shallow firn cores from Neumayer, Ekströmisen - A comparison of accumulation rates and stable isotope ratios
Archive:	Fedora at AWI
Author(s):	Schlosser, E.; Oerter, H.
Date:	2002-01-01
Subject(s):	Geo System; Structure and Dynamics of the Lithosphere and Polar Ice Sheets Neumayer, TEMPERATURE SENSORS
Description:	http://www.awi-bremerhaven.de/php/PublicationAbstracts/abstract.php?uid=Sch2002a
Publisher:	Annals of Glaciology, 35, 91-96, none, 2002
Type:	text, article
Rights:	http://creativecommons.org/licenses/by-nc-nd/2.0/legalcode
Abstract:	Since 1979/80 glaciological studies were carried out at Ekströmisen, Antarctica, including accumulation stake measurements, snow pit and shallow firn core studies. Snow stratigraphy, chemical properties, and stable isotope ratios (d18O, dD) were investigated. This study focuses on three cores taken between 1982 and 1998. Dating of the 1998-core was done using DEP, d18O-profiles and stake measurements. Accumulation rates show high interannual and spatial variability due to the extreme wind influence. No significant trend was found for the last 50 years, before that, accumulation decreased. The high spatial and interannual variability, however, should warn to be careful with trends. In spite of the highly irregular accumulation distribution, stable isotope ratios show little spatial variability. The mean annual d18O values of cores B04 and FB0198 agree fairly well for the time period from 1955 to 1982 covered by both cores. d18O values have increased during most of the 20th century; since the late 1980s a decrease is observed. This change is not related to air temperature, since mean annual air temperatures at Neumayer show no significant trend over the last two decades.

Abbildung 5.20: Demonstration: 1 – Details einer Publikation

Stellvertretend für andere Publikationen ist ein Ergebnis in Abb. 5.20 zu sehen.

Always give the proper citation when using this data!
Click here for citation details.

Record Details

Title:	Meteorological synoptical observations from Georg-von-Neumayer station, Antarctica during 1992
Archive:	Pangaea
Author(s):	König-Langlo, Gert
Date:	2005-05-23T15:42:57
Spatial Coverage:	
North:	-70.617
East:	-8.367
South:	-70.617
West:	-8.367
Mean:	
Latitude:	-70.617
Longitude:	-8.367
Elevation:	
min:	36.00 m
max:	36.00 m
Project(s):	Meteorological Long-Term Observations @ AWI (AWI_Meteo)
Event(s):	
Label:	**Georg-von-Neumayer**
Latitude:	-70.617
Longitude:	-8.367
Elevation:	36.00 m
DateTime:	1981-01-01T00:00:00
Latitude 2:	0.000
Longitude 2:	0.000
Elevation 2:	0.00 m
DateTime 2:	1992-12-31T00:00:00
Location:	Dronning Maud Land
CampaignName:	AWI_stations
Basis:	Neumayer Station
GearDevice:	Research station
GearType:	Multiple investigations
DOI:	doi:10.1594/PANGAEA.271598
Size:	8154 data points

Abbildung 5.21: Demonstration: 1 – Details der Datensätze, Teil 1

Citation: *König-Langlo, Gert (2005): Meteorological synoptical observations from Georg-von-Neumayer station, Antarctica during 1992, PANGAEA, doi:10.1594/PANGAEA.271598*

Parameter(s):

Parameter	Short Name	Unit	Label	Principal Investigator	Method	Comment
ALTITUDE	Altitude	m				Geocode
DATE/TIME	Date/Time					Geocode
LATITUDE	Latitude					Geocode
LONGITUDE	Longitude					Geocode
Whiteout yes/no	96111	y/n		König-Langlo, Gert	Visual observation	
High cloud	CH	code		König-Langlo, Gert	Visual observation	
Low cloud	CL	code		König-Langlo, Gert	Visual observation	
Middle cloud	CM	code		König-Langlo, Gert	Visual observation	
Total cloud amount	N	code		König-Langlo, Gert	Visual observation	
Low/middle cloud amount	Nh	code		König-Langlo, Gert	Visual observation	
Pressure, atmospheric	POPOPOPO	hPa		König-Langlo, Gert	Barometer	
Past blowing snow	S#8	code		König-Langlo, Gert	Visual observation	
Present blowing snow	S8	code		König-Langlo, Gert	Visual observation	
Temperature, air	TTT	deg C		König-Langlo, Gert	Thermometer	
Dew/frost point	TdTdTd	deg C		König-Langlo, Gert	Hygrometer	
Temperature, air, min	TnTnTn	deg C		König-Langlo, Gert	Thermometer	
Temperature, air, max	TxTxTx	deg C		König-Langlo, Gert	Thermometer	
Horizontal view distance	VV	code		König-Langlo, Gert	Visibility sensor	
Past weather1	W1	code		König-Langlo, Gert	Visual observation	
Past weather2	W2	code		König-Langlo, Gert	Visual observation	
Present weather	WW	code		König-Langlo, Gert	Visual observation	
Characteristic of barometric tendency	a	code		König-Langlo, Gert	Barometer	
Wind direction	dd	deg		König-Langlo, Gert	Anemometer	
Wind velocity	ff	m/sec		König-Langlo, Gert	Anemometer	
Cloud base height code	h	code		König-Langlo, Gert	Ceilometer	
Amount of barometric tendency	ppp	hPa		König-Langlo, Gert	Barometer	

Download dataset as MarineXML-file (in development, for testing purpose only)
Download dataset as tab-delimited textfile from Pangaea.de

view dataset

Attention: Large size (8154 data points) - maybe long loading time!

Abbildung 5.22: Demonstration: 1 – Details der Datensätze, Teil 2

In den Abbildungen 5.21 und 5.22 werden die Details eines „dataset“-Ergebnisses dargestellt. Das erste Bild zeigt die allgemeinen Daten zu der Messung, das zweite Bild die Liste mit den verwendeten Parametern. Unter dieser Liste befindet sich der Button „view dataset“, mit welchem der Benutzer sich die Messdaten tabellarisch darstellen kann.

Date/Time	Altitude [m]	h [code]	VV [code]	dd [deg]	ff [m/sec]	TTT [deg C]	TdTdTd [deg C]	POPOPOPO [hPa]	a [code]	ppp [hPa]
1992-01-01T00:00	36	9	99	120	5.1	-0.7	-1.7	991.5	1	0.7
1992-01-01T03:00	36			90	10.8	-0.6	-1.8	991.1	8	0.4
1992-01-01T06:00	36			90	11.3	-0.5	-9.6	992.0	3	0.9
1992-01-01T09:00	36	4	97	90	10.3	0.4	-0.8	993.0	2	1.0
1992-01-01T12:00	36	4	97	90	10.8	0.5	-0.7	993.6	2	0.6
1992-01-01T15:00	36	3	95	90	11.3	0.4	-0.7	994.2	2	0.6
1992-01-01T18:00	36	3	96	90	8.7	0.0	-1.2	994.9	2	0.7
1992-01-01T21:00	36	4	97	80	7.2	-0.5	-1.7	996.2	2	1.3
1992-01-02T00:00	36	4	97	100	7.7	-0.7	-1.9	997.2	2	1.0
1992-01-02T03:00	36			80	7.2	-1.5	-2.6	997.0	8	0.2
1992-01-02T06:00	36			90	4.1	-1.1	-2.2	997.1	3	0.1
1992-01-02T09:00	36	4	98	130	2.1	-0.7	-4.9	997.1	4	0.0
1992-01-02T12:00	36	4	98	160	2.6	0.5	-5.1	997.0	8	0.1
1992-01-02T15:00	36	6	98	170	1.0	0.1	-6.9	996.3	7	0.7
1992-01-02T18:00	36	5	98	220	1.0	-0.1	-6.9	995.9	6	0.4

Abbildung 5.23: Demonstration: 1 – Auszug aus den Messdaten

Ein Ausschnitt aus einer solchen, meist sehr umfangreichen Tabelle ist in Abb. 5.23 dargestellt.

Der Benutzer möchte nun diese Messdaten in einem universellem Format abspeichern und wählt den entsprechenden Link, der ihm eine MarineXML-Datei generiert. Ein Auszug dieser Datei wurde bereits in Abbildung 5.11 auf Seite 80 dargestellt.

5.5.2.2 Szenario 2: Erweiterte Suche

Das zweite Szenario geht von dem Wunsch des Benutzers aus, eine detailliertere Suche durchzuführen. Die entsprechende Seite wird dabei von der Startseite aus aufgerufen (s. Abb. 5.24).

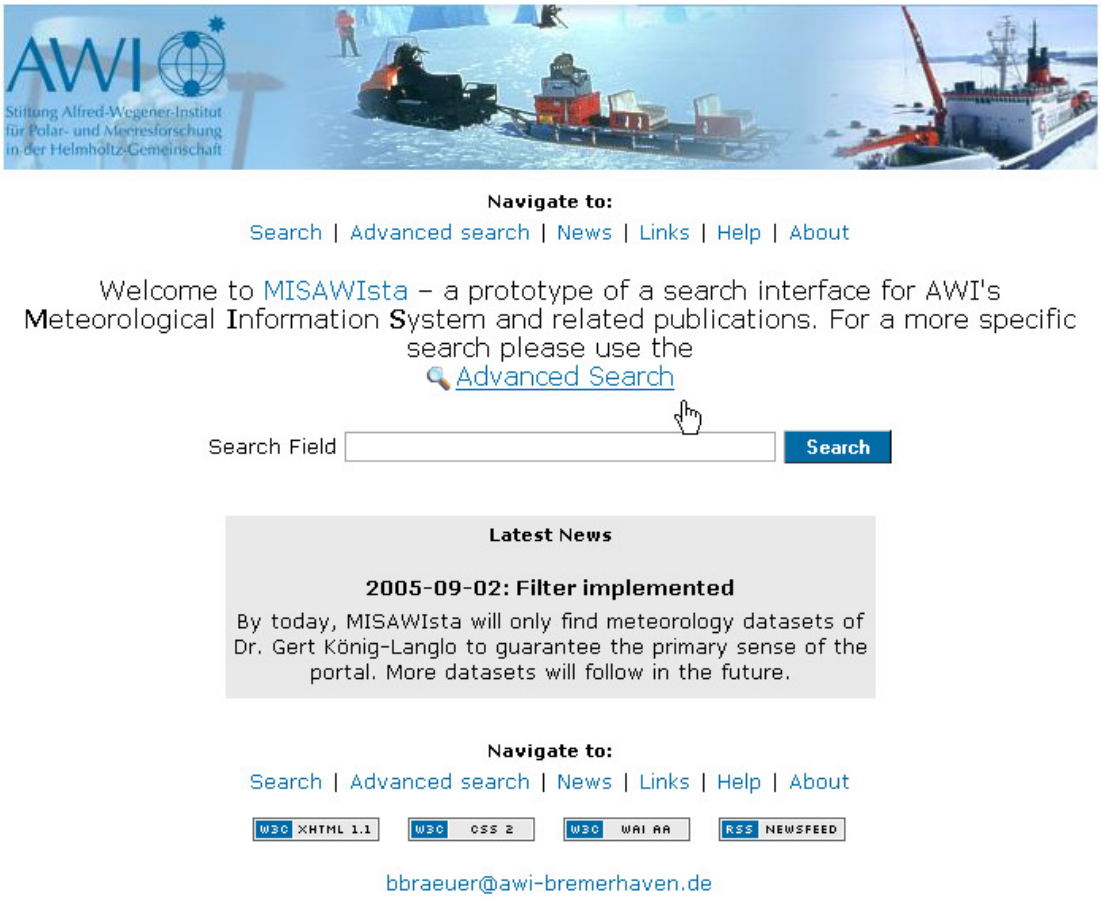


Abbildung 5.24: Demonstration: 2 – Aufruf der Startseite / Auswahl „Advanced Search“

Nun besteht für den Benutzer die Möglichkeit, zwischen drei Sucharten zu wählen:

. . . freie Suche mit Koordinaten

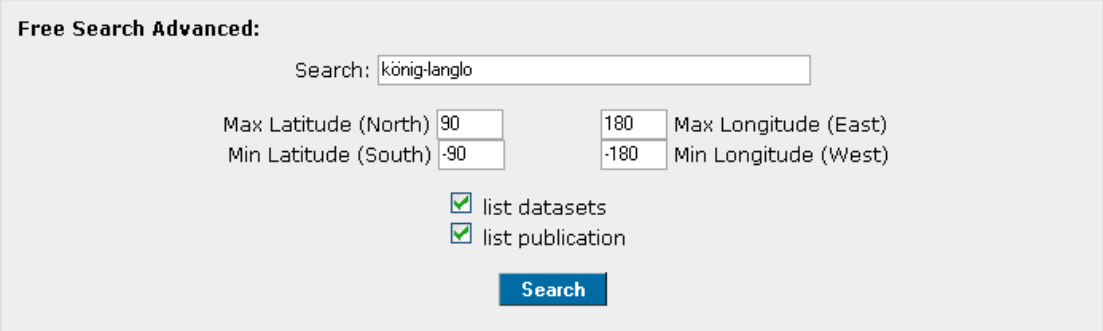


Abbildung 5.25: Demonstration: 2 – Freie Suche mit Koordinaten

Der Benutzer hat die Möglichkeit, eine freie Suche durchzuführen und sich dabei auf ein bestimmtes Gebiet zu beschränken. Die in Abbildung 5.25 gemachten Einstellungen umfassen eine weltweite Suche, das Ergebnis wäre dasselbe wie bei der Verwendung der einfachen Suche.

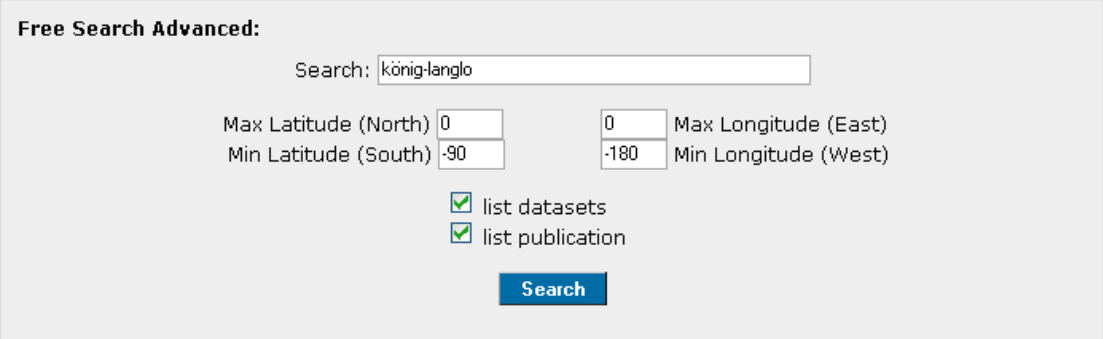


Abbildung 5.26: Demonstration: 2 – Freie Suche mit geänderten Koordinaten

In Abbildung 5.26 hat der Benutzer die Koordinaten geändert und sucht nun nur nach Datensätzen, welche sich westlich des Nullmeridians in der südlichen Hemisphäre befinden.

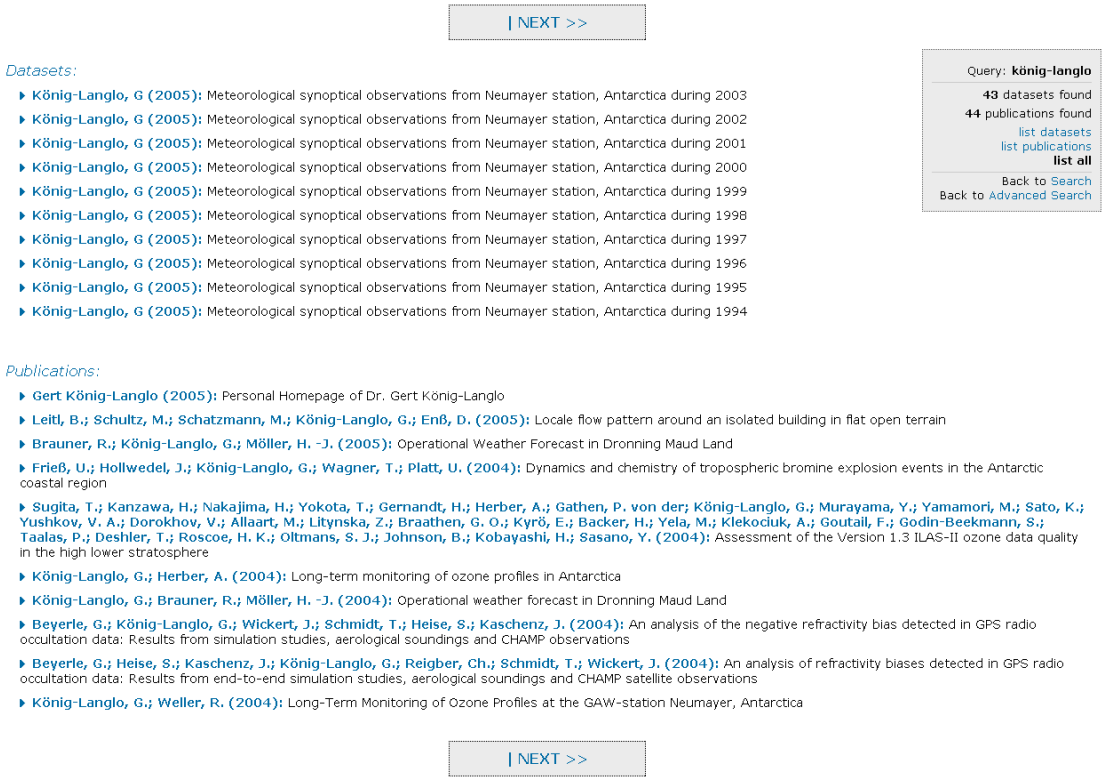


Abbildung 5.27: Demonstration: 2 – Ergebnis der freien Suche

Die Ergebnisse (Abb. 5.27) sind hier weniger als bei einer weltweiten Suche. Zu beachten ist: In den Publikationsdaten sind keine Koordinaten vermerkt, daher werden alle zu dem Suchbegriff passenden Ergebnisse geliefert.

. . . Suche nach Schiffdaten

Als zweite Möglichkeit bietet sich dem Nutzer die Suche nach einer Reise eines Forschungsschiffs, in diesem Beispiel die „Polarstern“.

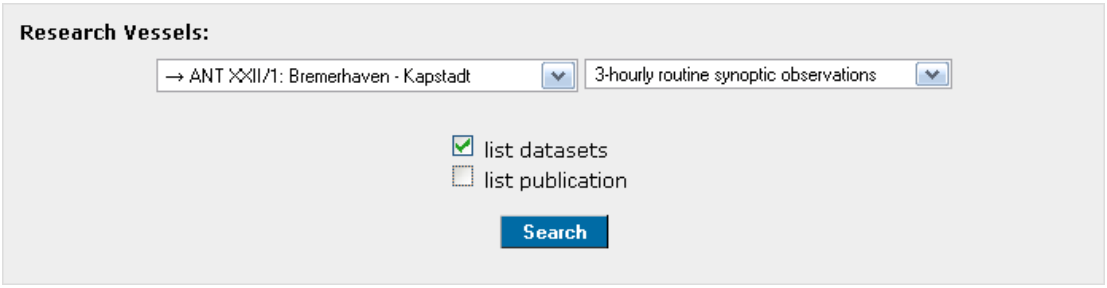


Abbildung 5.28: Demonstration: 2 – Suche nach Schiffsdaten

Der Benutzer möchte hier Daten zu dem ersten Fahrtabschnitt der Reise XXII von Bremerhaven nach Kapstadt (s. Abb. 5.28). Ferner möchte er nur Datensätze

angezeigt bekommen, für die Publikationen interessiert er sich nicht – daher wird die entsprechende Checkbox abgewählt.

Das Ergebnis ist genau ein Datensatz, dargestellt in Abbildung 5.29.

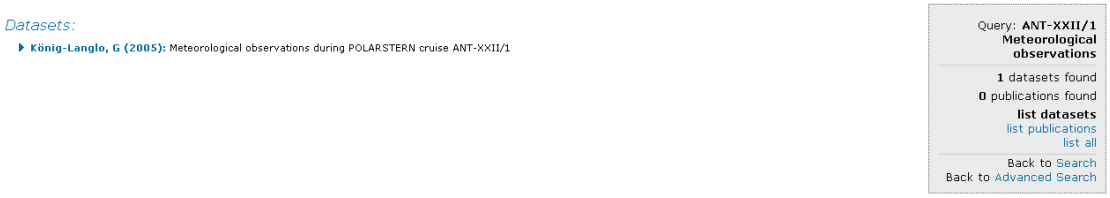


Abbildung 5.29: Demonstration: 2 – Ergebnis der Suche nach Schiffsdaten

. . . Suche nach Stationsdaten

Die dritte Suchoption ermöglicht dem Benutzer die Suche nach Stationsdaten (s. Abb. 5.30). Hier soll nach Daten der Koldewey-Station des Jahres 2000 gesucht werden.

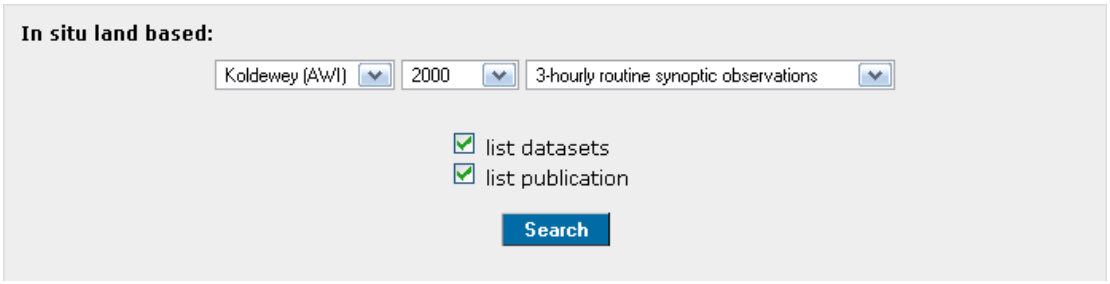


Abbildung 5.30: Demonstration: 2 – Suche nach Stationsdaten

Das Ergebnis zeigt die Abbildung 5.31. Das Bild zeigt außerdem, dass für diesen speziellen Suchstring keine Publikationen existieren.

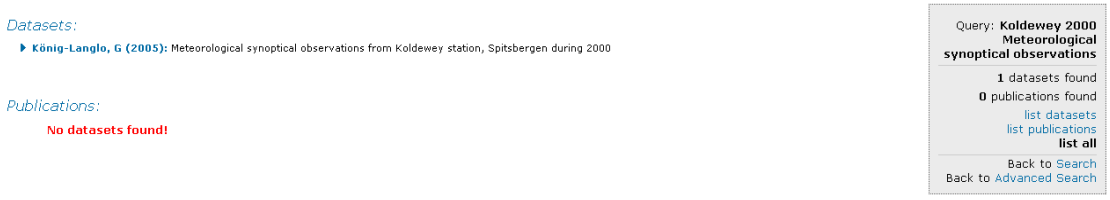


Abbildung 5.31: Demonstration: 2 – Ergebnis der Suche nach Stationsdaten

5.5.2.3 Szenario 3: Fehler in einem System

Der dritte Fall beschäftigt sich mit dem Eintreten eines Fehlers. Die Abbildung 5.32 zeigt die Ausgabe des Portals, wenn der Dienst nicht erreicht werden konnte. Der Benutzer wird aufgefordert, es noch einmal zu probieren oder bei wiederholtem Auftreten den Entwickler zu kontaktieren.

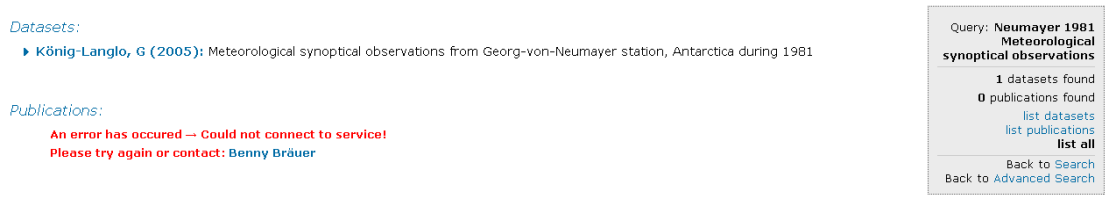


Abbildung 5.32: Demonstration: 3 – Dienst nicht erreichbar

Ein weiterer Fehler – oder besser gesagt: eine Information – ist die Notwendigkeit eines Logins bei Pangaea, um die (dort) befindlichen Messdaten anzuzeigen (s. Abb. 5.33). Möchte der Benutzer die Daten eines unveröffentlichten Projekts ansehen, ist eine Anmeldung bei Pangaea Pflicht. Das Portal selbst zeigt nur die frei zugänglichen Daten.

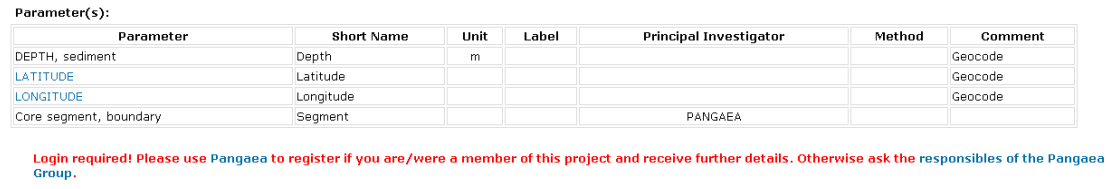


Abbildung 5.33: Demonstration: 3 – Messdaten benötigen Anmeldung

5.6 Weiterführende Planungen

Mit der Demonstration des funktionstüchtigen Prototypen ist die Entwicklung abgeschlossen worden. Dennoch gibt es wie bei jeder anderen Softwareentwicklung auch das „Problem“, dass immer neue Ideen für Erweiterungen, Verbesserungen, Änderungen usw. an dem System existieren. Hat man aber ein Projekt angefangen, sollte man es auch im vorgegebenen zeitlichen Rahmen abschließen, obgleich man noch neue Ideen einbringen könnte. Diese können immer noch über ein Nachfolgeprojekt oder im Rahmen der Wartung eingepflegt werden.

Einige dieser Ideen sind nun in diesem Abschnitt erläutert. Es handelt sich dabei um Anregungen, welche die Entwicklung betreffen, Verbesserungen, welche man einbauen könnte, geplante Änderungen, welche zukünftige Versionen von Erweiterungen mitbringen etc. Die Aussicht für die zukünftige Entwicklung, sprich Änderungen, welche die Grundlagen des Systems betreffen werden in Kapitel 6 ausführlicher behandelt.

5.6.1 Einführung eines Session-Management

Vom jetzigen Standpunkt aus ist die Seite als nicht besonders hoch frequentiert zu betrachten. Mit der Einführung weiterer Web Services anderer Institute etc. sollte allerdings ein umfangreiches Session-Management eingefügt werden.

Sessions in PHP bieten die Möglichkeit, bestimmte Daten während einer Folge von Abrufen der Webseiten zu speichern. Dadurch können die Anwendungen persönlicher gestaltet werden. Beim Aufruf der Seite erhält der Benutzer eine eindeutige ID, die so genannte Session-ID, welche in einem „Cookie“ gespeichert oder in der URL abgelegt wird.

Die Session-Verwaltung sollte dabei Unterstützung von einer MySQL-Datenbank erfahren. Das Ablegen der ID's sowie der Daten, deren Umfang durch die Einbettung weiterer Web Services stark zunehmen wird, kann so weitaus effizienter erfolgen. Der Abruf der Daten und das Nutzen der Variablen ist damit einfacher, als es mit der von PHP unterstützten Methode der Fall ist.

Auf dieses umfangreiches Session-Management wurde aber innerhalb der Diplomarbeit verzichtet. Die Begründung liegt darin, dass die Zukunft der Webseite in dieser „freien“ Form ungewiss ist, sondern die Einbindung in eine andere Umgebung erfolgen kann bzw. wird. Ferner verfügt der Prototyp noch über keine Funktionen, welche eine Speicherung „persönlicher“ Einstellungen notwendig machen. Die weitere Begründung dieses Vorgehens ist in der Sektion 6.2 erläutert.

Die Session-Technologie wurde dennoch verwendet, um die Übergabe von Variablen von einem Skript zum nächsten zu realisieren sowie den „Navigator“ und die „Sidebar“ zu implementieren. Diese hätten sich ohne Sessions nur mit unnötig großem Mehraufwand entwickeln lassen.

5.6.2 Bereitstellen weiterer Datenformate zum Herunterladen der Messdaten

Zur Zeit existieren zwei Auswahlmöglichkeiten bei der Speicherung der Messdaten. Zum Einen die von Pangaea gelieferte CSV-Datei, zum Anderen die von MISAWIsta generierte MarineXML-Datei. Zur weiteren Verarbeitung sind diese Formate mehr als ausreichend. Wieder Bezug auf das Kapitel 6 nehmend, werden zukünftig aber auch andere Formate notwendig sein.

So möchte der Nutzer vielleicht die Daten der Tabelle nicht als solche haben, sondern als ein Diagramm. Dank des XML-Formates (dessen aktuelle Form dafür nur leicht angepasst werden muss) und XSLT lassen sich die Daten in jede beliebige Form bringen, was den Bedarf an anderen Formaten deckt. Eine Überlegung wert

ist auf jeden Fall die Darbietung einer PDF-Datei, welche die Publikationsdaten, Parameter, Messdaten und ggf. Abbildungen in einer Datei zusammenfasst. Probleme können hier die Messdaten bereiten, da sie meistens über die normale Seitenbreite hinausgehen. Dies ist in zukünftige Überlegungen mit einzubeziehen.

6 Zukünftige Entwicklung / Aussicht

6.1 Data Warehouse

6.1.1 Grundlegendes

In fast allen Unternehmen und Institutionen existieren Unmengen an Daten, und mit jedem Tag wächst dieser Datenberg an. Das Problem ist, die Daten lagern meist in verschiedenen Systemen, den sog. „transaktionsorientierten Informationssystemen“, also den betrieblichen Produktiv- bzw. Arbeitssystemen. Diese bieten oft nur eingeschränkte Auswertungsmöglichkeiten, weil sie für die Bearbeitung großer Mengen gleichartiger Vorgänge und Daten ausgerichtet sind. Komplexe Abfragen, wie sie oft benötigt werden, verschlechtern meist die Leistung dieser Systeme, was den betrieblich Ablauf immens beeinträchtigen kann. Zudem lagern die Daten verteilt, was wiederum eine umfangreiche Abfrage zusätzlich erschwert.[1]

Einem Data Warehouse (Datenlager) liegen demnach zwei Hauptgedanken zugrunde. Zum Einen müssen die Daten aus verteilten und unterschiedlich strukturierten Datenbeständen integriert werden, um so eine Gesamtansicht auf die Quelldaten (und u.a. damit eine umfassende Auswertung) zu ermöglichen. Zum Anderen muss eine Trennung der Daten des operativen Geschäfts von denen erfolgen, welche – im Data Warehouse gelagert – dem Berichtswesen, der Entscheidungsunterstützung, der Geschäftsanalyse, des Controlling und der Unternehmensführung dienen.

Aus diesen Gedanken lassen sich nun die Anforderungen ein Data Warehouse ableiten[2]:

- einfacher und effizienter Zugriff auf die Daten des Unternehmens / des Instituts
- Daten müssen konsistent zur Verfügung stehen
- anpassungsfähig und flexibel gegenüber Änderungen

[1] vgl. [IWI05]
[2] vgl. [Kimball u. Ross 2002, S. 3ff]

- Sicherheit der teilweise hochsensiblen Daten gewährleisten
- Grundlage zum Schutz vor unbedachten / falschen Entscheidungen
- Data Warehouse muss von allen akzeptiert und konsequent angewendet werden, um den Hauptgedanken zu genügen

Unter einem Data Warehouse versteht man eine zentrale Datensammlung. Inhalt dieser Datensammlung sind Daten aus unterschiedlichen Quellen. Aus diesen Quellen werden die Daten in das Datenlager kopiert. Diesen Vorgang nennt man ETL-Prozess: extract, transform, load – Extraktion, Transformation, Laden. Daten werden aus verschiedenen Quellen extrahiert, durch Transformation bereinigt und vereinheitlicht sowie am Prozessende in das Datenlager geladen. Da dieser Prozess turnusgemäß durchgeführt werden kann, lassen sich die Daten im Data Warehouse auch nach zeitlichen Aspekten, also langfristig halten. Dies bietet die Möglichkeit, Analysen über die Zeit hinweg zu erstellen. Dabei lassen sich oft nicht nur die Vergangenheit, sondern auch zukünftige Änderungen (Marktschwankungen, Klimaänderungen etc.) analysieren.[3]

In Data Warehouses werden Daten speziell für Auswertungszwecke zusammengestellt, welche die Unternehmensleitung / Wissenschaftler bei ihren Entscheidungen oder Aussagen unterstützen. Dafür werden Auswertungstechniken wie *Online Analytical Processing* (OLAP) oder *Data Mining* verwendet. In Abbildung 6.1 ist das Prinzip hinter dem Data Warehouse dargestellt.

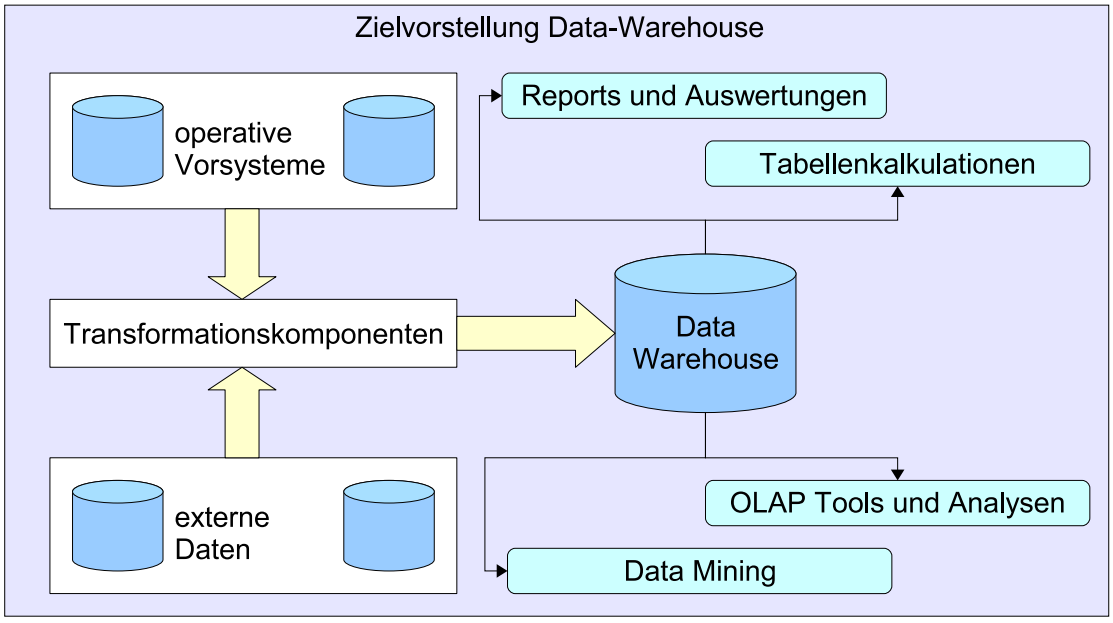


Abbildung 6.1: Aufbau des Data Warehouse-Konzepts, vgl. [Wiki05 b]

[3] vgl. [Wiki05 c]

Data Warehouses sind spezielle Datenbanken. Für ihren Zweck und der Art ihres Einsatzes benötigen sie allerdings bestimmte Funktionalitäten und vor allem eine besondere Architektur. Dies wird von gängigen Datenbanksystemen nicht unterstützt, was Softwarehersteller dazu veranlasst hat, spezialisierte Produkte anzubieten, die den Anforderungen besser entsprechen.

Ein solches Produkt ist „Sybase IQ“ der Firma Sybase. Basis der Software ist eine relationale Datenbank, welche auf Data Warehousing zugeschnitten wurde. Damit lassen sich sehr gute Raten bei der Datenkompression (bis weit in den Terabyte-Bereich) und höchst performante Abfragen (auch bei zahlreichen gleichzeitig stattfindenden Zugriffen) erreichen.

Sybase IQ kommt bei Pangaea zum Einsatz. Ziel ist es, in Voraussicht auf das C3-Grid-Projekt (s. S. 109ff) eine Grundlage für die massiven Steigerung von (komplexen) Datenbankabfragen zu schaffen. Dabei muss das System nicht nur die Ergebnisausgabe innerhalb eines vertretbaren Zeitraums realisieren können, sondern auch die Abfragen und Analysen technisch ermöglichen.

Die Abbildung 6.2 zeigt den konzeptionellen Aufbau von Sybase IQ. Die Daten aus den Produktivsystemen gelangen über die Verarbeitung im ETL-Prozess in die Speicherbereiche des Data Warehouse. Die „Datenbank“ von Sybase IQ stellt dabei die Verbindung zwischen den gespeicherten Daten und der Analyse-, Report- und Data Mining-Werkzeugen dar, die ihrerseits die Informationen automatisch oder nach Abruf an die Benutzer weiterleiten.

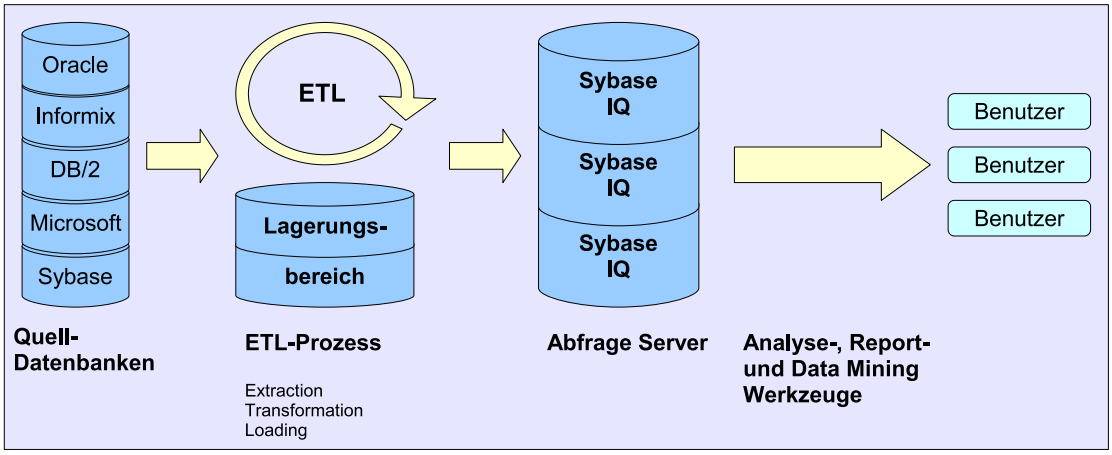


Abbildung 6.2: Aufbau des Sybase IQ-Konzepts, vgl. [Khosroschahli 2004, S. 6]

Diese Werkzeuge können u.a. in Form eines (oder mehrerer) Web Service zur Verfügung gestellt werden, um den Nutzern den Zugriff auf das Data Warehouse zu ermöglichen. Dies bietet sich bei Pangaea besonders an, da das System bei der Datenabfrage ohnehin auf Web Services setzt und daher die benötigten Strukturen für den Einsatz von Web Services bereits vorhanden sind.

6.1.2 Verwendung in MISAWIsta

Bei Fertigstellung des Data Warehouse bei Pangaea wird wie erwähnt ein neuer Web Service bereitgestellt bzw. der bestehende erweitert werden. Über diesen Dienst erfolgt dann wiederum der Zugriff auf die im Pangaea Data Warehouse gespeicherten Daten. Es genügt also, eine neue Verbindung zu dem Web Service in den Quelltext von MISAWIsta einzubauen, um die neuen Möglichkeiten und Funktionen eines Data Warehouses zu nutzen.

Bietet der Web Service die ihm z.Z. angedachten Funktionen, wird ein selektiver Zugriff auf die Messdaten ermöglicht. Sie stehen dann für benutzerdefinierte Operationen zur Verfügung. Die Möglichkeiten von MISAWIsta können dadurch weit reichend gesteigert werden, zumindest im Bezug auf die Messdaten. Der Benutzer des Systems erhält eine weitere Suchmöglichkeit: Die Suche nach meteorologischen Messdaten, also den Werten selbst. Dies kann z.B. durch ein Formular geschehen, welches in Auswahllisten mit verschiedenen, an eine SQL-Struktur angelehnte, Einträge enthält. Diese müssen dabei so formuliert sein, dass sie auch von einem Nutzer ohne besondere Kenntnisse der EDV intuitiv genutzt werden können.

Die Spannweite der Abfrage sollte sich dabei nicht nur auf einen bestimmten Datensatz beschränken, welcher die jeweiligen Messdaten beinhaltet, sondern übergreifend sein. So kann eine sinnvolle Nutzung des Data Warehouse (vom Standpunkt der Benutzer gesehen) erfolgen. Denn viele Nutzer, nicht nur aus dem Bereich der Wissenschaft, sondern auch der Wirtschaft, wie z.B. Reedereien könnten sich für die Abfrage auf mehrere Messreihen interessieren. Zum Beispiel eine Übersicht der Windverhältnisse der letzten 10 Jahre am berüchtigten Kap Horn. Solche Informationen sind für die Reedereien von immenser Wichtigkeit und können, richtig verwendet, dabei helfen, die Gefährdung der Schiffe auf See oder auch die Kosten zu senken. Diese Art von Anfragen werden des Öfteren an die Wissenschaftler des AWI gestellt, welche wiederum ihre eigenen Arbeitsdatenbanken befragen mussten, um die Antwort zu liefern. Durch die Speicherung der Daten in Pangaea und dem Einsatz des Data Warehouse in Verbindung mit einem Client wie MISAWIsta sind diese Anfragen und die damit einhergehende Bindung von Personal und Ressourcen nicht mehr notwendig. Stattdessen können sich Interessenten mittels graphischer Benutzeroberfläche selbst die Informationen besorgen (ob die Daten bei entsprechendem Umfang bzw. Einsatz maschineller Ressourcen völlig kostenlos angeboten werden, sei an dieser Stelle dahingestellt).

Nichtsdestotrotz bieten sich durch das Data Warehouse-Konzept zahlreiche neue Möglichkeiten, auch solche, die sich erst mit der Entwicklung des Web Service herauskristallisieren werden. Doch mit dem Data Warehouse ist für MISAWIsta noch lange nicht Schluss. Der nächste Schritt ist die mögliche Integration das C3-Grid-Projekt.

6.2 C3 – Grid

6.2.1 Grid-Computing

Grid-Computing umfasst alle Methoden zur Zusammenfassung von Rechenleistung zahlreicher Computer innerhalb eines Netzwerks, welche dadurch die parallele Lösung von rechenintensiven Problemen ermöglichen (Stichwort: *verteiltes Rechnen*). Das Grid-Konzept sieht vor, dass diese Rechenleistung bei Bedarf von anderen Ressourcen abgerufen werden kann.

Grid's (deutsch: Gitter) werden vor allem dort eingesetzt, wo enorme Rechenleistung erforderlich ist – z.B. bei der Auswertung und Darstellung von sehr großen Datenmengen aus der medizinischen oder der meteorologischen Forschung. Anwender, die diese Leistung benötigen, können sie über eine bestimmte Software (z.B. ein Portal) erhalten, in dem sie über die Software ihr Abfrage / Rechenoperation dem Grid übergeben und es über dessen Dienste berechnen lassen.

Dieses Prinzip des verteilten Rechnens sowie die Nutzung meist brachliegender Rechenkapazität verringern die immensen Anschaffungskosten im Bereich Hoch- und Höchstleistungsrechner in einem gewissen Umfang. Die Gesamtkosten des Betriebes eines solchen Supercomputers („total cost of ownership“) lassen sich ebenfalls gering halten, da Wartung und Personal nicht in einem derartigen Umfang notwendig werden, wie das bei Supercomputern der Fall ist. Durch die sehr gute Skalierbarkeit von Grid's jedoch können diese Superrechner auch in ein Grid eingebunden werden und so die Gesamtleistung enorm erhöhen.

Großes Ziel bzw. zur Zeit noch Vision der Grid-Entwicklung ist das „computing grid“, welches nach dem Prinzip des „power grid“ arbeitet – dem Stromnetz. Ein Benutzer verbindet sich über eine bestimmte Software (die „Steckdose“) mit dem Grid. Was im Grid selbst passiert, bleibt dem Nutzer verborgen. Er nutzt die im Grid dargebotene Leistung für seine Rechenoperation(en) und beendet nach dem Abschluss dieser die Sitzung. Handelt es sich um ein kommerzielles Grid, kann für die verbrauchte Leistung eine Bezahlung verlangt werden – genau wie beim Stromnetz.

Zu den bekanntesten Grid-Projekten, an denen sich jeder Besitzer eines PC's o.ä. beteiligen kann, zählen u.a. *SETI@home*[4] (Search for ExtraTerrestrial Intelligence at home – die Suche nach außerirdischer Intelligenz in radioastronomischen Daten) und *Folding@home*[5], bei welchem die Faltung von Proteinen berechnet wird.

Um noch einen Leistungsvergleich zu bringen: SETI@home hat ca. eine halbe Million aktive Nutzer (insgesamt über 5 Millionen), die zur Zeit eine Gesamtrechenleistung von über 200 TeraFLOPS (floating-point operations per second)

[4] vgl. http://setiathome.ssl.berkeley.edu/
[5] vgl. http://folding.stanford.edu/

erbringen. Der zur Zeit schnellste Supercomputer der Welt[6], der BlueGene/L von IBM, bringt es auf ca. 135 TeraFLOPS.

6.2.2 Projekt

C3-Grid steht für *Collaborative Climate Community Data and Processing Grid*. Es ist Bestandteil der „D-Grid"-Initiative des BMBF. Das Ziel des C3-Projekts ist die Entwicklung einer hochproduktiven Grid-basierten Umgebung für die deutsche Erdsystemforschungsgemeinschaft zur effektiven wissenschaftlichen Analyse von hochvolumigen Erdsystemmodell- und -beobachtungsdaten sowie deren verteiltem Zugriff, Transport und Konsistenzsicherung.[7]

Ein typischer Workflow eines Nutzers aus der Erdsystemforschung unterteilt sich dabei in drei Teile (vgl. Abb. 6.3): Zuerst werden Basisdaten zusammengetragen und vorbereitet. Es folgt die Aufbereitung und Auswertung der Datensätze für die spezifische wissenschaftlichen Anwendung. Am Ende steht dann die Darstellung der Ergebnisse.[8]

Bei den Workflows kann es sich dabei u.a. um folgende handeln:

- eine regionale Klimasimulation auf Basis globaler Modelldaten
- die Diagnose verschiedener globaler Modellsimulationen
- die Begleitung von Messkampagnen

[6] vgl. http://top500.org/lists/plists.php?Y=2005&M=06
[7] vgl. [Hiller u. Fritzsch 2005, S. 3]
[8] vgl. [Hiller u. Fritzsch 2005, S. 11ff]

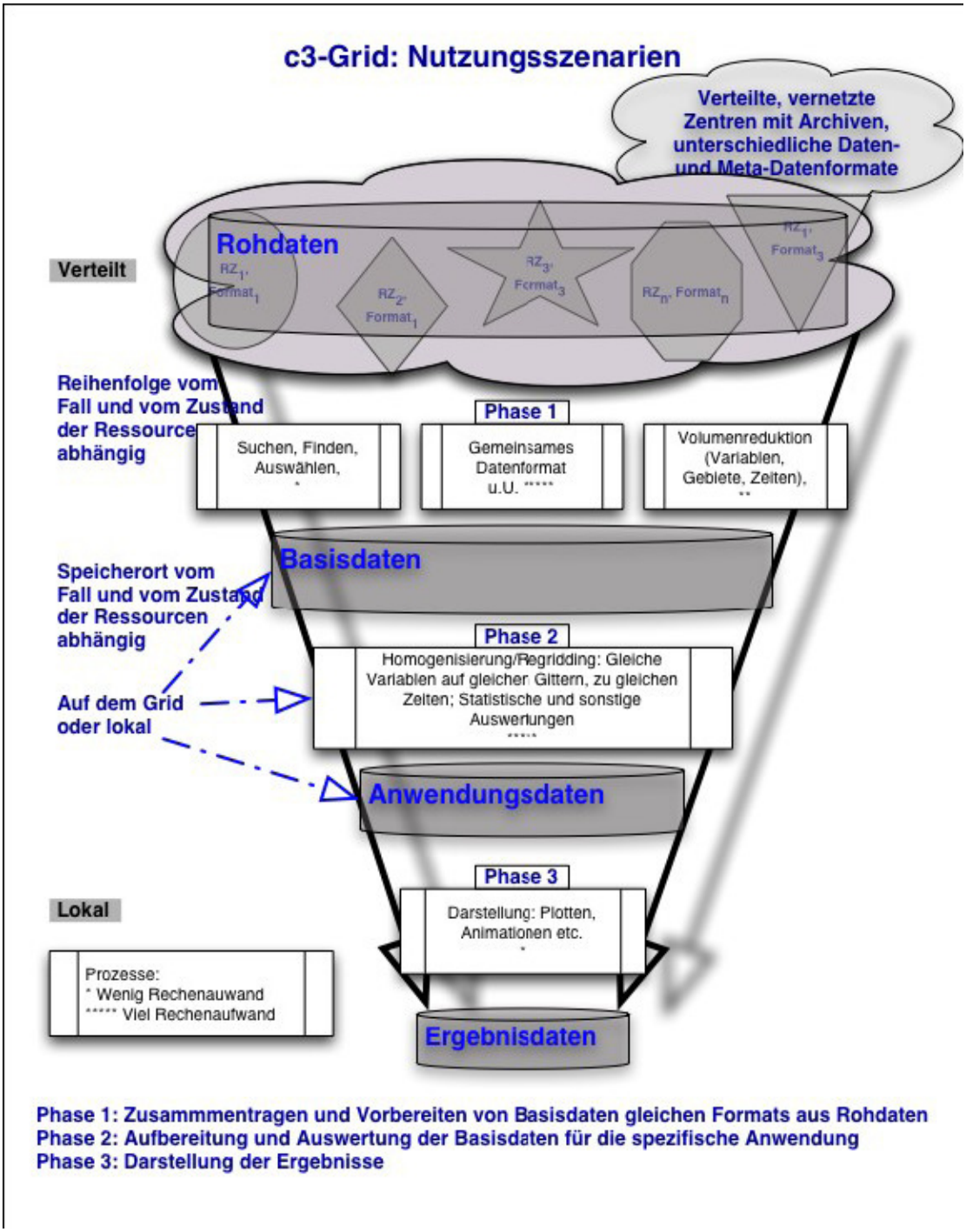


Abbildung 6.3: Schematische Darstellung eines Workflows, vgl. [Hiller u. Fritzsch 2005, Abb. 3, S. 12]

6.2.3 Verwendung von MISAWIsta

Ein Teilziel des C3-Projekts ist die Schaffung eines Portals, um den Benutzern die Möglichkeit zu geben auf die Datenbestände, welche das Grid zur Verfügung stellt zuzugreifen. Die Software für das Portal wird das JSR-168-kompatible[9] Open Source Projekt *GridSphere*[10] sein. Meldet sich ein Nutzer über dieses Portal an, soll er verschiedene Optionen, wie die Abfrage auf Daten oder – in einer späteren Phase des Projekts – Rechenzeit auf Hoch- und Höchstleistungsrechnern buchen können.

Web Services können in einem Grid zahlreiche Verwendung finden. Ein Beispiel (schematisch in Abb. 6.4 dargestellt) soll dies verdeutlichen: Ein Client stellt eine Anfrage an einen Service. Dieser führt nun verschiedene Operationen durch. Zuerst überprüft er die Benutzerdaten des Anwenders auf ihre Korrektheit (1). Ist diese gegeben kann der Benutzer die gewünschte Operation ausführen. Der Web Service, welcher die Operation abfängt (2), holt parallel von drei Datenanbietern Informationen bzw. Daten (3). Diese liefert der Dienst an zwei weitere Web Services (Intermediaries), die ihrerseits verschiedene Berechnungen anstellen (4).

Nach Abschluss der Rechenoperation gelangen die Ergebnisse über SOAP zurück an den Ausgangs-Web Service. Nun hat der Nutzer eine Ausgabe der Daten als Diagramm gewünscht, sowie das Herunterladen einer PDF-Datei. Der Wunsch wird an zwei weitere Dienste gesendet, welche die Formatierungen vornehmen und diese dann z.B. im XSLT-Format zurückschicken (5). Das Endergebnis geht nun an das Portal (6), welches wiederum die letzten Operationen (Transformation der XML-Daten nach PDF bzw. Diagramm-Bild) durchführt und dem Benutzer zur Verfügung stellt.[11]

[9] Java Specification Request

[10] vgl. http://www.gridsphere.org/

[11] Aufgrund der Web Service-Bezogenheit der Arbeit wurde hier mehr auf diesen Aspekt eingegangen und die parallel, verteilten Berechnungen, welche in einem Grid stattfinden, außen vorgelassen.

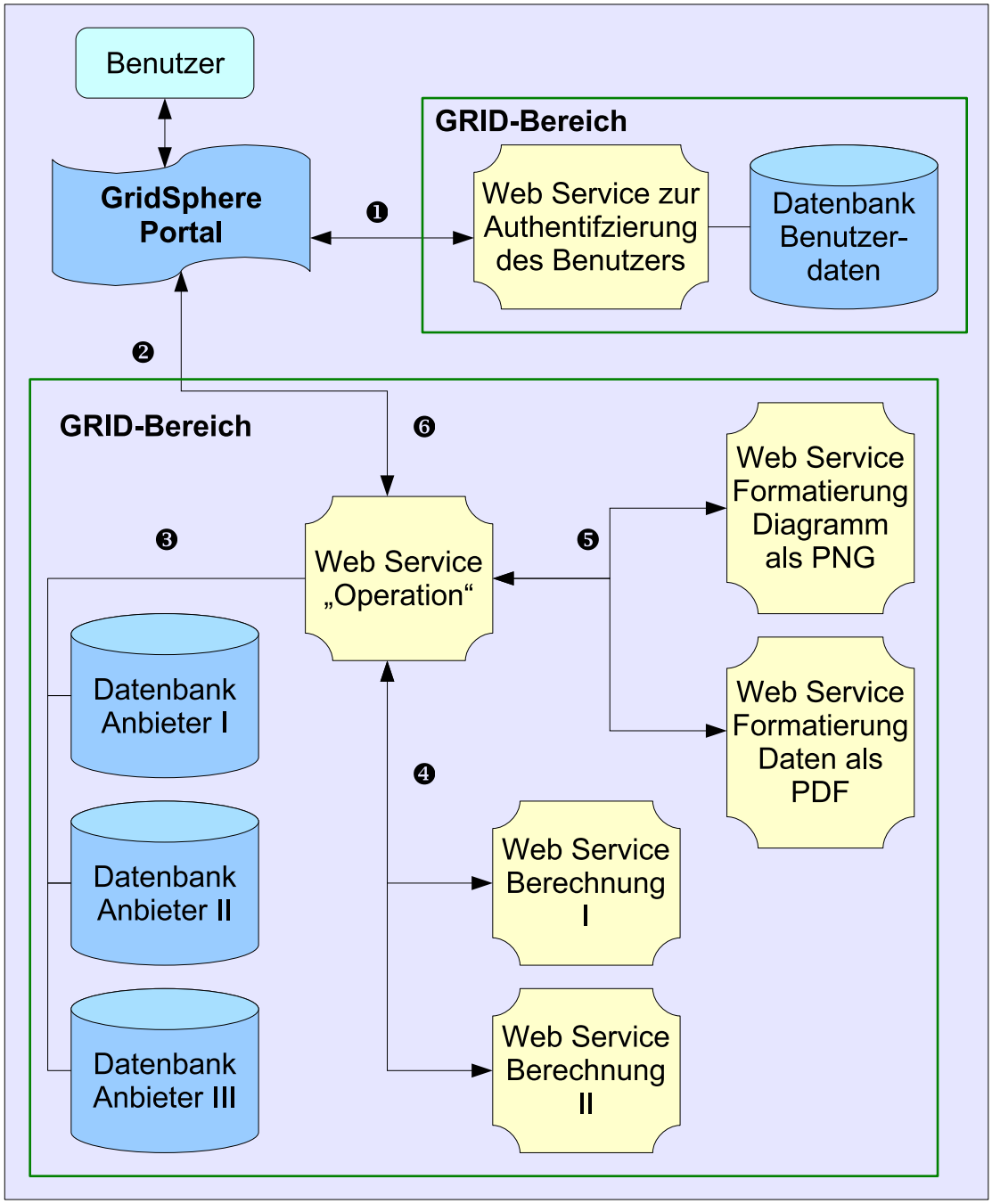


Abbildung 6.4: Schematische Darstellung einer Grid-Sitzung

Der Verwendung von Web Services sind dabei kaum Grenzen gesetzt. Ein Problem könnte die enorme Datenmenge sein, welche übertragen wird. Doch dafür stellt das Grid die passende Infrastruktur zur Verfügung, da die Datenlast aufgeteilt werden kann. Der Benutzer selbst bekommt dann nur jene Daten geliefert, welche er zuvor angefragt hat.

MISAWIsta kann durch seine spezielle Ausrichtung eine Grundlage für den meteorologischen Abschnitt des Portals darstellen. Die Suchfunktionen können beliebig erweitert und für die Systeme, auf denen ein Zugriff erfolgt, verfeinert werden. Der Transport in eine höhere Programmiersprache, wie Java beim GridSphere-Portal, kann aufgrund der einfach gehaltenen Struktur ohne größeren Entwicklungsaufwand erfolgen, sollte dies notwendig sein.

Für die Ergebnisse, welche Messdaten enthalten, soll der operative Zugriff auf diese Daten möglich sein. Das heißt, der Benutzer sieht nicht eine (oder mehrere) riesige Tabellen sondern wird sich, über ein einfaches Interface eine Abfrage zusammenbauen können. „Selektiere die gemessenen Temperatur- und Luftdruck-Jahresmittel für die Jahre zwischen 1985 und 2005 in der Antarktis und stelle das Ergebnis als Diagramm auf dem Bildschirm sowie als PDF-Datei zur Verfügung“ könnte ein solcher Suchbefehl lauten, dem Abfragen auf die verschiedenen Informationssysteme und Datenbank der im Grid befindlichen Institutionen folgen. Ein anderer Dienst kümmert sich dann um die Formatierung der Daten für die Darstellung.

7 Resümee

Zum Abschluss des Projekts „Diplomarbeit“ möchte ich ein Resümee geben. Die Anstrengungen von vier Monaten resultieren in der vorliegenden Arbeit. Nicht immer ist Umsetzung der zentralen Aufgabe einfach gewesen: war eine Hürde beseitigt, trat an anderer Stelle eine neue auf. Doch nun gehören die Probleme der Vergangenheit an. Die Anwendung ist vollendet und die letzten Zeilen der Dokumentation sind niedergeschrieben.

Keine andere Arbeit während des Studiums hat einen solchen Arbeits- und Leistungsaufwand erforderlich gemacht. Die Anstrengungen aber haben sich dennoch gelohnt. Denn der Nutzen, welcher aus diesem praxisorientierten Projekt entsteht, ist für meine fachliche Zukunft nicht zu ersetzen.

Viele der während der Ausbildung behandelten Themen konnten nutzbringend eingesetzt werden: Objektorientierte Programmierung, die Anwendung der Web Service-Architektur, Umgang mit UNIX, Software-Engineering und -modellierung, Qualitätssicherung, Ergonomie von Software und Webanwendungen, Management eines Projekts etc. bis hin zu den Erfahrungen im Verfassen wissenschaftlicher Arbeiten.

All diese Bereiche flossen in diese Arbeit ein. Herausgekommen ist ein Prototyp, der nur darauf wartet, immer weiterentwickelt und mit neuen Funktionen und Technologien bestückt zu werden. Es freut mich zu sehen, wenn dieses Portal weiter Verwendung findet – vielleicht nicht in dieser Form – es aber so einen kleinen Beitrag zur Vereinfachung der komplexen Datenwelt leisten kann.

Benny Bräuer
Midlum, den 29. September 2005

Abkürzungsverzeichnis

ANT	Antarktis
API	Application Programming Interface
ARK	Arktis
AWI	Alfred-Wegener-Institut für Polar- und Meeresforschung
Blob	Binary Large Object
BMBF	Bundesministerium für Bildung und Forschung
C3-Grid	Collaborative Climate Community Data and Processing Grid
CIP	Catalogue Interoperability Protocol
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CSV	Character Separated Values
CTRL	Control
DBlib	Database Library
DC	Dublin Core
DCOM	Distributed Component Object Model
de	deutsch
DIF	Directory Interchange Format
div	division
DOI	Digital Object Identifier
DOM	Document Object Model
DTD	Dokument Type Definition
EDV	Elektronische Datenverarbeitung
EFTS	Electronic Full-Text Sources
en	englisch
ePIC	electronic Publication Information Center
EPK	Ereignisgesteuerte Prozesskette
ETL	Extract, Transform, Load bzw. Extraktion, Transformation, Laden
FAQ	Frequently Asked Questions
Fcntl	File Control (System)
Fedora	Flexible (and) Extensible Digital Object Repository Architecture

FGDC	Federal Geographic Data Committee
FLOPS	Floating-point Operations Per Second
FTP	File Transfer Protocol
GB	Gigabyte
GCMD	Global Change Master Directory
GHz	Gigahertz
GIMP	GNU Image Manipulation Program
GNU	GNU is not Unix
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifikation
IEEE	Institute of Electrical and Electronic Engineers
ISBN	International Standard Book Number
ISSN	International Standard Serial Number
IT	Informationstechnik, auch Informationstechnologie
Java RMI	Java Remote Method Invocation
JSR	Java Specification Request
MarCoPolI	Marine, Coast, Polar, Infrastruktur
MB	Megabyte
MHz	Megahertz
MIME	Multipurpose Internet Mail Extensions, auch Multimedia Internet Message Extensions
MIS	Meteorology Information System
NM	Neumayer
NY	Ny-Ålesund
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OASIS	Organization for the Advancement of Structured Information Standards
OLAP	Online Analytical Processing
PC	Personal Computer
PDA	Personal Digital Assistant
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
PID	Persistent Identifier
RDF	Resource Description Framework
RFC	Requests for Comments
RPC	Remote Procedure Call
RSS	Really Simple Syndication
RTF	Rich Text Format
SAML	Security Assertion Markup Language
SETI	Search for Extraterrestrial Intelligence
SGML	Standard Generalized Markup Language

SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SP1	Service Pack 1
SQL	Structured Query Language
STRG	Steuerung
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Ver	Version
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WS	Web Service(s)
WSDL	Web Service Description Language
WSRP	Web Services for Remote Portlets
XAMPP	X Apache MySQL Perl PHP
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XPath	XML Path Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformation
XT	Extreme Tailoring

Literaturverzeichnis

AWI05 a
Alfred-Wegener-Institut für Polar- und Meeresforschung (Hrsg.): *Das Alfred-Wegener-Institut.* http://www.awi-bremerhaven.de/AWI/index-d.html. – Online–Ressource, Abruf: 16. Juli 2005

AWI05 b
Alfred-Wegener-Institut für Polar- und Meeresforschung (Hrsg.): *Das Alfred-Wegener-Institut für Polar- und Meeresforschung.* http://www.awi-bremerhaven.de/AWI/organigramm-d.html. – Online–Ressource, Abruf: 19. Juli 2005

Balzert 1999a
Balzert, Heide: *Lehrbuch der Objektmodellierung: Analyse und Entwurf.* Heidelberg : Spektrum, Akademischer Verlag, 1999 (Lehrbücher der Informatik)

Balzert 1999b
Balzert, Helmut: *Lehrbuch Grundlagen der Informatik.* Heidelberg : Spektrum, Akademischer Verlag, 1999 (Lehrbücher der Informatik)

CELab05
Carl von Ossietzky Universität Oldenburg (Hrsg.): *Dublin Core Metadata (DC).* http://www.celab.de/elearning_know_how/dc.html. – Online–Ressource, Abruf: 18. Juli 2005

CM05
F&P GmbH - FEiG & PARTNER (Hrsg.): *Darf's vielleicht auch Freeware sein?* http://www.contentmanager.de/magazin/artikel_586_freeware_open_source_cms.html. – Online–Ressource, Abruf: 23. Juli 2005

Fed05
Fedora Project (Hrsg.): *Major Features of the Fedora Repository.* http://www.fedora.info/download/2.0/userdocs/server/features/features.html. – Online–Ressource, Abruf: 24. Juli 2005

Graser 2005
GRASER, Franz: Skriptsprache strickt Services zusammen. In: *Computer Zeitung* 32-33 (2005), August, S. 14

Hiller u. Fritzsch 2005
HILLER, Prof. Dr. W. ; FRITZSCH, Dr. B.: *Projektantrag C3-Grid*. : Alfred-Wegener-Institut für Polar- und Meeresforschung, Mai 2005. – Projektantrag für das C3-Grid (Community-Projekt zur Erdsystem-/Klimaforschung)

IWI05
INSTITUT FÜR WIRTSCHAFTSINFORMATIK, UNIVERSITÄT BERN (Hrsg.): *Data Warehouse*. `http://www.ie.iwi.unibe.ch/forschung/km/datawarehouse.php`. – Online–Ressource, Abruf: 11. August 2005

Khosroschahli 2004
KHOSROSCHAHLI, Kia: Sybase IQ - Schneller und sicherer Zugriff auch bei exponentiellem Datenwachstum. In: *Gipfeltreffen Hochperformance/RZ-Virtualisierung/Security* best Systeme GmbH, 2004

Kimball u. Ross 2002
KIMBALL, Ralph ; ROSS, Margy ; ELLIOTT, Robert (Hrsg.): *The Data Warehouse Toolkit*. Second Edition. New York : Wiley Computer Publishing - John Wiley and Sons, Inc., 2002

Langner 2003
LANGNER, Torsten: *Web Services mit Java - Neuentwicklung und Refactoring in der Praxis*. München : Markt+Technik Verlag, 2003 (new technology)

Ncr05
NETCRAFT LTD. (Hrsg.): *June 2005 Web Server Survey*. `http://news.netcraft.com/archives/2005/06/01/june_2005_web_server_survey.html`. – Online–Ressource, Abruf: 18. Juli 2005

Onken 2005
ONKEN, Bastian: *Prototypentwicklung und ökonomischer Leistungsvergleich von OAI- und Web Service-orientierten Architekturen zur Realisierung von Repositories für Publikationen in einer wissenschaftlichen Großforschungseinrichtung*, Hochschule Bremerhaven, Diplomarbeit, Oktober 2005

Pan05 a
ZENTRUM FÜR MARINE UMWELTWISSENSCHAFTEN, ALFRED-WEGENER-INSTITUT FÜR POLAR- UND MEERESFORSCHUNG (Hrsg.): *PANGAEA - ein Informationssystem für marine Umweltdaten*. `http://www.pangaea.de/Info/paper/PANGAEA_Info.html`. – Online–Ressource, Abruf: 31. Juli 2005

Pan05 b
ZENTRUM FÜR MARINE UMWELTWISSENSCHAFTEN, ALFRED-WEGENER-INSTITUT FÜR POLAR- UND MEERESFORSCHUNG (Hrsg.): *Struktur der Metadaten*. `http://ws.pangaea.de/xml/MetaData.png`. – Online–Ressource, Abruf: 10. August 2005

SH05
SELFHTML E.V. (Hrsg.): *Sinn und Zweck von Stylesheets*. `http://de.selfhtml.org/css/intro.htm`. – Online–Ressource, Abruf: 20. Juli 2005

Stein 2004
STEIN, Sebastian: *Emergenz in der Softwareentwicklung – bereits verwirklicht oder Chance?*, Hochschule für Technik und Wirtschaft Dresden, Diplomarbeit, Mai 2004

Sybase 1999
SYBASE INC. (Hrsg.): *Benutzerhandbuch für Sybase® Adaptive Server™ Enterprise Transact-SQL®*. Oktober 1999

Vbip01
VBIP (Hrsg.): *SOAP Messages*. `http://www.vbip.com/books/1861005091/chapter_5091_02.asp`. – Online–Ressource, Abruf: 01. August 2005

W3C CSS 2005
W3C (Hrsg.): *Cascading Style Sheets, level 2 revision 1*. Version: Juni 2005. `http://www.w3.org/TR/CSS21/`. – Online–Ressource, Abruf: 18. August 2005

W3C SOAP 2003
W3C (Hrsg.): *SOAP Version 1.2 Part 1: Messaging Framework*. Version: Juni 2003. `http://www.w3.org/TR/soap12/`. – Online–Ressource, Abruf: 18. August 2005

W3C WSDL 2003
W3C (Hrsg.): *Web Services Description Language (WSDL) Version 1.2*. Version: März 2003. `http://www.w3.org/TR/2003/WD-wsdl12-20030303/`. – Online–Ressource, Abruf: 18. August 2005

W3C XHTML 2001
W3C (Hrsg.): *XHTML™1.1 - Module-based XHTML*. Version: Mai 2001. `http://www.w3.org/TR/xhtml11`. – Online–Ressource, Abruf: 18. August 2005

W3C XML 2004
W3C (Hrsg.): *Extensible Markup Language (XML) 1.1*. Version: Februar

2004. http://www.w3.org/TR/xml11/. – Online–Ressource, Abruf: 18. August 2005

W3C02
W3C (Hrsg.): *Web Services Architecture Requirements - W3C Working Draft 11 October 2002*. http://www.w3.org/TR/2002/WD-wsa-reqs-20021011#IDAGWEBD. – Online–Ressource, Abruf: 01. August 2005

W3C04
W3C (Hrsg.): *Extensible Markup Language (XML) 1.1*. http://edition-w3c.de/TR/2004/REC-xml11-20040204/. – Online–Ressource, Abruf: 19. Juli 2005

Wiki05 a
Wikimedia Foundation Inc. (Hrsg.): *Cascading Style Sheets*. http://de.wikipedia.org/wiki/CascadingStyleSheets. – Online–Ressource, Abruf: 20. Juli 2005

Wiki05 b
Wikimedia Foundation Inc. (Hrsg.): *Data-Warehouse*. http://de.wikipedia.org/wiki/Bild:Datawarehouse.png. – Online–Ressource, Abruf: 11. August 2005

Wiki05 c
Wikimedia Foundation Inc. (Hrsg.): *Data-Warehouse*. http://de.wikipedia.org/wiki/Data_Warehouse. – Online–Ressource, Abruf: 11. August 2005

Wiki05 d
Wikimedia Foundation Inc. (Hrsg.): *Digital Object Identifier*. http://de.wikipedia.org/wiki/Digital_Object_Identifier. – Online–Ressource, Abruf: 18. Juli 2005

Wiki05 e
Wikimedia Foundation Inc. (Hrsg.): *Extensible Hypertext Markup Language*. http://de.wikipedia.org/wiki/Xhtml. – Online–Ressource, Abruf: 20. Juli 2005

Wiki05 f
Wikimedia Foundation Inc. (Hrsg.): *Meteorologie*. http://de.wikipedia.org/wiki/Meteorologie. – Online–Ressource, Abruf: 18. Juli 2005

Wiki05 g
Wikimedia Foundation Inc. (Hrsg.): *Perl*. `http://de.wikipedia.org/wiki/Perl`. – Online–Ressource, Abruf: 19. Juli 2005

Wiki05 h
Wikimedia Foundation Inc. (Hrsg.): *PHP*. `http://de.wikipedia.org/wiki/Php`. – Online–Ressource, Abruf: 19. Juli 2005

Wiki05 i
Wikimedia Foundation Inc. (Hrsg.): *Web service*. `http://en.wikipedia.org/wiki/Image:Webservices.png`. – Online–Ressource, Abruf: 01. August 2005

Wiki05 j
Wikimedia Foundation Inc. (Hrsg.): *XML*. `http://de.wikipedia.org/wiki/Xml`. – Online–Ressource, Abruf: 19. Juli 2005

A Quelltexte

A.1 export.pl

```
#!/usr/local/bin/perl

  # Das Exportscript fuer Daten der Meteorologie aus Sybase (nach Pangaea) wurde
      als Teil einer Diplomarbeit von Benny Braeuer (bbraeuer@awi-bremerhaven.de,
       b.braeuer@gmx.net) erstellt.
  # Change Log:
    # Version 1.1 - Stand 29.07.2005
      # einige Operationen als Subroutinen in den Footer ausgelagert
      # Ueberfluessigen Quelltext entfernt
      # Kommentare vervollstaendigt
    #   Version 1.00 - Stand 01.06.2005

  # ---------------------------------------------------------------------
  #SCRIPT-HEADER
  # ---------------------------------------------------------------------

  # Verschiedene Bibliotheken einbinden
  use Term::ReadKey;
  use Sybase::DBlib;
  use Fcntl;

  # Pfad fuer nicht in Perl enthaltene Erweiterungen setzen
  use lib "/mobs1/user1/misawi/pangaea/lib/";
  use MIME::Lite;

  # Server festlegen
  $srv = "AWI";

  # Benoetigt fuer die Readline-Verarbeitung
  open(IN,"</dev/tty");
  *OUT = *IN;

  # Hier wird der Name und der Pfad der Exportdatei gesetzt
  $export_dat_pfad = "/mobs1/user1/misawi/pangaea/";

  # ---------------------------------------------------------------------
  #SCRIPT-BODY
  # ---------------------------------------------------------------------

  # Auswahl des Datenursprungs
  DATENURSPRUNGSAUSWAHL:
    print "\nBitte Datenursprung wählen:\n";
    print "\t1 - Polarstern\n";
    print "\t2 - Neumayer-Station\n";
    print "\t3 - Koldewey-Station\n";
    $datenursprung = ReadLine 0;
    chop $datenursprung;

```

```
47  # Funktionsaufruf
48  datenursprung($datenursprung);
49
50  #  Abfrage von Benutzername und Passwort
51  DBDATEN:
52    ReadMode 1, IN;
53
54    print "\nGeben Sie bitte Ihren Benutzernamen ein:\n";
55    $uid = ReadLine 0;
56    chop $uid;
57
58    print "Geben Sie bitte Ihr Benutzerpasswort ein:\n";
59    ReadMode('noecho');
60    $pwd = ReadLine 0;
61    chop $pwd;
62
63    ReadMode 0, IN;
64
65  # ------------------------------------------------------------------
66
67  # Aufbau der Datenbankverbindung
68  $export = new Sybase::DBlib $uid, $pwd, $srv or goto DBDATEN;
69
70  # Funktionsaufruf
71  dbabfragen($datenursprung);
72
73  # SQL-Abfrage ausfuehren
74  $export->dbsqlexec;
75
76  # ------------------------------------------------------------------
77
78  # Variable fuer die Kontrollausgabe
79  $zeile = 2;
80
81  # Abfrageresultat in die Datei schreiben
82  while($export->dbresults != NO_MORE_RESULTS)
83  {
84    # jede Zeile kommt in ein Array "@dat"
85    while(@dat = $export->dbnextrow)
86    {
87      # Alle Strings der Liste ...
88      foreach $export (@dat)
89      {
90        # ...in die Datei schreiben, Tabulator getrennt
91        print DATEI "$export\t";
92      }
93
94      # Kontrollausgabe
95      print "Schreibe Zeile: $zeile\n";
96      $zeile++;
97
98      # Zeilenumbrauch nach jedem geschriebenem Datensatz
99      print DATEI "\n";
100   }
101
102   # Textausgabe nach Abschluss des Schreibens
103   if ($datenursprung eq "1")
104   {
105     print "Schreiben von <<$reiseziel-$reisenr/$fahrtabschnitt>> - Daten
          erledigt!\n";
106   }
107   elsif ($datenursprung eq "2")
108   {
```

```
109       print "Schreiben von Neumayer-Daten erledigt!\n";
110     }
111     elsif ($datenursprung eq "3")
112     {
113       print "Schreiben von Koldewey-Daten erledigt!\n";
114     }
115
116     # Datei wird abgeschlossen
117     close(DATEI);
118   }
119   # Datenbankverbindung wird geschlossen
120   $export->dbclose;
121
122   # --------------------------------------------------------------------
123
124   # Stringoperationen
125   open (ALTEDATEI, $datei);
126   while (<ALTEDATEI>)
127   {
128     # Y/N gibts derzeit nur bei Neumayerdaten (Whiteout)
129     if ($datenursprung eq "2")
130     {
131       # Suchen und Ersetzen
132       $suchen = "\tJ\t";
133       $ersetzen = "\tY\t";
134
135       # Operation
136       s/$suchen/$ersetzen/;
137     }
138
139     push (@NEUEDATEI,$_);
140   }
141   close (ALTEDATEI);
142
143   open (NEUEDATEI, ">$datei");
144   for (@NEUEDATEI)
145   {
146     print NEUEDATEI $_;
147   }
148   close (NEUEDATEI);
149
150   # --------------------------------------------------------------------
151
152   # Moeglichkeit, die Exportdatei direkt per Mail an den Importeur (Datenkurator)
        zu schicken.
153   # MIME::Lite-Paket unter http://www.zeegee.com
154   # MIME::Lite Dokumentation unter http://www.zeegee.com/code/perl/MIME-Lite/docs
      /
155   MAIL:
156     print "Soll die erstellte Datei als Mail verschickt werden? (j/n)\n";
157     $mailjn = ReadLine 0;
158     chop $mailjn;
159
160   # --------------------------------------------------------------------
161
162   # Als Mail versenden?
163   if ( $mailjn eq "j" or $mailjn eq "J" )
164   {
165     print "Wie lautet die Adresse des Empfaengers? (z.B. hgrobe\@awi-bremerhaven.
          de)\n";
166     $empfaenger = ReadLine 0;
167     chop $empfaenger;
168
```

```
169     $betreff = "Neue Exportdaten fuer Pangaea";
170
171     # Variabler Teil des Nachrichtentextes
172     if ($datenursprung eq "1")
173     {
174       $text = "die Polarsternreise $reiseziel-$reisenr/$fahrtabschnitt";
175     }
176     elsif ($datenursprung eq "2")
177     {
178       $text = "die Neumayer-Station von $zeitraum1 und $zeitraum2";
179     }
180     elsif ($datenursprung eq "3")
181     {
182       $text = "die Koldewey-Station von $zeitraum1 und $zeitraum2";
183     }
184
185     # Der Nachrichtentext
186     $ntext1 = "Hallo Hannes,\n\n";
187     $ntext2 = "anbei die neuen Daten fuer $text zum Importieren nach Pangaea.\n\n
            ";
188     $ntext3 = "Liebe Grueße, Gert";
189
190     # Mailinhalte bearbeiten
191     # Funktionsaufruf
192     Mail_File(  "gkoenig\@awi-bremerhaven.de",
193           "$empfaenger",
194           "$betreff",
195           "$ntext1$ntext2$ntext3",
196           "BINARY",
197           "$export_dat_pfad$datei",
198           "$datei");
199   }
200
201   # --------------------------------------------------------------------
202   #SCRIPT-FOOTER
203   # --------------------------------------------------------------------
204
205   # Welche Daten sollen exportiert werden
206   sub datenursprung
207   {
208     # Parameter einlesen
209     my $datenursprung = shift;
210
211     # Polarstern
212     if ($datenursprung eq "1")
213     {
214       $messort = 2;
215       $hoehe = "25";
216
217       REISEZIEL:
218       print "Bitte Reiseziel eingeben (z.B. ANT oder ARK)\n";
219       $reiseziel = ReadLine 0;
220       chop $reiseziel;
221
222       # Falsches Reiseziel abfangen
223       if ($reiseziel eq "ANT" or $reiseziel eq "ARK")
224       {}
225       else
226       {
227         goto REISEZIEL;
228       }
229
230       print "Bitte Reisenummer eingeben (z.B. XXII)\n";
```

```
231       $reisenr = ReadLine 0;
232       chop $reisenr;
233
234       print "Bitte Fahrtabschnitt eingeben (z.B. 1 oder 4b)\n";
235       $fahrtabschnitt = ReadLine 0;
236       chop $fahrtabschnitt;
237     }
238
239     # Neumayer
240     elsif ($datenursprung eq "2")
241     {
242       $messort = "(1,4,10,11,17,18,19,20,21)";
243       $sname = "Neumayer";
244       $shoehe = "40";
245
246       print "Bitte Anfangszeit eingeben (z.B. 2007-01-01 00:00)\n";
247       $zeitraum1 = ReadLine 0;
248       chop $zeitraum1;
249
250       print "Bitte Endzeit eingeben (z.B. 2007-12-31 21:00)\n";
251       $zeitraum2 = ReadLine 0;
252       chop $zeitraum2;
253     }
254
255     # Koldewey
256     elsif ($datenursprung eq "3")
257     {
258       $messort = "(3)";
259       $sname = "Koldewey";
260       $shoehe = "11";
261
262       print "Bitte Anfangszeit eingeben (z.B. 2007-01-01 00:00)\n";
263       $zeitraum1 = ReadLine 0;
264       chop $zeitraum1;
265
266       print "Bitte Endzeit eingeben (z.B. 2007-12-31 21:00)\n";
267       $zeitraum2 = ReadLine 0;
268       chop $zeitraum2;
269     }
270
271     #... alle anderen Eingaben
272     else
273     {
274       goto DATENURSPRUNGSAUSWAHL;
275     }
276   }
277
278   sub dbabfragen
279   {
280     # Parameter einlesen
281     my $datenursprung = shift;
282
283     # Polarstern
284     if ( $datenursprung eq "1" )
285     {
286       #Abfrage
287       $export->dbcmd("select  'Track.$reiseziel-$reisenr/$fahrtabschnitt' as '
              Event label',
288                 convert(numeric(5,1), GeoBreite) as '1600',
289                 convert(numeric(5,1), GeoLaenge) as '1601',
290                 '$hoehe' as '4607',
291                 convert (char(26), DatumUhrzeit, 109) as '1599',
292                 Wolkenuntergrenze as '45259',
```

```
293                     HorSicht as '45260',
294                     Windrichtung as '2221',
295                     convert(numeric(5,1), Windgeschw) as '18906',
296                     convert(numeric(5,1), Temperatur) as '4610',
297                     convert(numeric(5,1), Taupunkt) as '4611',
298                     convert(numeric(5,1), Luftdruck) as '2224',
299                     ArtLuftdruckAenderung as '45311',
300                     convert(numeric(5,1), BetragLuftdruckAenderung) as '45312',
301                     GegenwWetter as '45261',
302                     VergWetter1 as '45262',
303                     VergWetter2 as '45263',
304                     TiefeWolken as '45264',
305                     MittlereWolken as '45265',
306                     HoheWolken as '45266',
307                     GesamtBedeckung as '45267',
308                     BedeckungClCm as '45268',
309                     convert(numeric(5,1), MaxTemperatur) as '5151',
310                     convert(numeric(5,1), MinTemperatur) as '5150',
311                     GegenwSchneetreiben as '45307',
312                     VergSchneetreiben as '45308',
313                     Whiteout as '45309',
314                     GemKurs as '45257',
315                     GemGeschw as '45258',
316                     convert(numeric(5,1), Wassertemperatur) as '717',
317                     PeriodeWindsee as '18959',
318                     HoeheWindsee as '45269',
319                     RichtungDuenung1 as '45270',
320                     HoeheDuenung1 as '45271',
321                     PeriodeDuenung1 as '45290',
322                     RichtungDuenung2 as '45272',
323                     HoeheDuenung2 as '45273',
324                     PeriodeDuenung2 as '45291',
325                     Eisansatz as '45274',
326                     StaerkeEisansatz as '45310',
327                     ZuAbEis as '45275',
328                     MeereisChr as '45276',
329                     EisEntwicklungsZustand as '45277',
330                     LandeisImMeer as '45278',
331                     EisrandRichtung as '45279',
332                     Eissituation as '45280'
333
334                 from  MetDB.dbo.ObseDat o,
335                     MisawiDB.dbo.Reisen r
336                 where o.Messort_ID# = $messort
337                 and   r.Reise = '$reiseziel $reisenr'
338                 and   r.Fahrtabschnitt = '$fahrtabschnitt'
339                 and   o.DatumUhrzeit between r.Abfahrtsdatum and r.Ankunftsdatum
340                 order by o.DatumUhrzeit");
341
342         # Anlegen der Exportdatei
343         $datei = "exp-PS.$reiseziel-$reisenr-$fahrtabschnitt";
344
345         # Funktionsaufruf
346         fileexist($datei);
347         open(DATEI,">>$datei");
348
349         # Pangaea-ID's in die Datei schreiben
350         # VORSICHT - Stets mit groesster Sorgfalt behandeln!
351         print DATEI "Event label\t1600\t1601\t4607\t1599\t45259\t45260\t2221\t18906
                \t4610\t4611\t";
352         print DATEI "2224\t45311\t45312\t45261\t45262\t45263\t45264\t45265\t45266\
                t45267\t45268\t";
```

```
353       print DATEI "5151\t5150\t45307\t45308\t45309\t45257\t45258\t717\t18959\
              t45269\t45270\t45271\t";
354       print DATEI "45290\t45272\t45273\t45291\t45274\t45310\t45275\t45276\t45277\
              t45278\t45279\t45280\n";
355     }
356
357     # Abfrage fuer Neumayer und Koldewey
358     elsif ($datenursprung eq "2" or $datenursprung eq "3")
359     {
360       # Abfrage
361       $export->dbcmd("select  '$sname' as 'Event label',
362                   '$shoehe' as '4607',
363                   convert (char(26), DatumUhrzeit, 109) as '1599',
364                   Wolkenuntergrenze as '45259',
365                   HorSicht as '45260',
366                   Windrichtung as '2221',
367                   convert(numeric(5,1), Windgeschw) as '18906',
368                   convert(numeric(5,1), Temperatur) as '4610',
369                   convert(numeric(5,1), Taupunkt) as '4611',
370                   convert(numeric(5,1), Luftdruck) as '2224',
371                   ArtLuftdruckAenderung as '45311',
372                   convert(numeric(5,1), BetragLuftdruckAenderung) as '45312',
373                   GegenwWetter as '45261',
374                   VergWetter1 as '45262',
375                   VergWetter2 as '45263',
376                   TiefeWolken as '45264',
377                   MittlereWolken as '45265',
378                   HoheWolken as '45266',
379                   GesamtBedeckung as '45267',
380                   BedeckungClCm as '45268',
381                   convert(numeric(5,1), MaxTemperatur) as '5151',
382                   convert(numeric(5,1), MinTemperatur) as '5150',
383                   GegenwSchneetreiben as '45307',
384                   VergSchneetreiben as '45308',
385                   Whiteout as '45309'
386
387               from  MetDB.dbo.ObseDat
388               where Messort_ID# in $messort
389               and   DatumUhrzeit between '$zeitraum1' and '$zeitraum2'
390               order by DatumUhrzeit");
391
392       # Zerlege Datum und Uhrzeit, speichere in Array
393       @zeitraum1 = split(/ /, $zeitraum1);
394       @zeitraum2 = split(/ /, $zeitraum2);
395
396       # Anlegen der Exportdatei
397       # fuer Neumayer
398       if ($datenursprung eq "2")
399       {
400         $datei = "exp-NM.@zeitraum1[0]_@zeitraum2[0]";
401
402         #Funktionsaufruf
403         fileexist($datei);
404         open(DATEI,">>$datei");
405       }
406       # fuer Koldewey
407       elsif ($datenursprung eq "3")
408       {
409         $datei = "exp-NA.@zeitraum1[0]_@zeitraum2[0]";
410
411         # Funktionsaufruf
412         fileexist($datei);
413         open(DATEI,">>$datei");
```

```
414       }
415
416       # Pangaea-ID's in die Datei schreiben
417       # VORSICHT - Stets mit groesster Sorgfalt behandeln!
418       print DATEI "Event label\t4607\t1599\t45259\t45260\t2221\t18906\t4610\t4611
              \t2224\t45311\t45312\t";
419       print DATEI "45261\t45262\t45263\t45264\t45265\t45266\t45267\t45268\t5151\
              t5150\t45307\t45308\t45309\n";
420     }
421
422     # Rechte setzen (664 = Lese- und Schreibrechte fuer Eigner und Gruppe, sowie
          Leserechte fuer alle)
423     chmod(0664,$datei);
424   }
425
426   # Funktion -  Variablen mit den Mailinhalten / Absenden
427   sub Mail_File
428   {
429     # Parameter einlesen
430     my $myMailAddress =$_[0];   # Absender
431     my $email_address =$_[1];   # Empfaenger
432     my $title         =$_[2];   # Betreff
433     my $body_message  =$_[3];   # Nachrichtentext
434     my $fileType      =$_[4];   # Art des Anhangs (gaengigerweise 'BINARY' oder '
          TEXT')
435     my $fileName      =$_[5];   # Pfad+Dateiname des Anhangs
436     my $outFileName   =$_[6];   # Dateiname
437
438     # Erstelle MIME::Lite Mail Objekt
439     my $msg = MIME::Lite->new
440     (
441       From     => $myMailAddress,
442       To       => $email_address,
443       Subject  => $title,
444       Type     => 'multipart/mixed',
445     );
446
447     $msg->attach
448     (
449       Type     => 'TEXT',
450       Data     => $body_message
451     );
452
453     $msg->attach
454     (
455       Type        => $fileType,
456       Path        => $fileName,
457       Filename    => $outFileName
458     );
459
460     # Mail abschicken
461     if($msg->send())
462     {
463       print "Mail an $empfaenger verschickt!\n"
464     }
465     else
466     {
467       print "Fehler: Mail nicht verschickt!\n";
468       goto MAIL;
469     }
470   }
471
```

```
472   # Funktion - Fragt ab, ob eine Datei existiert, und falls ja, ob sie geloescht
          werden soll
473   sub fileexist()
474   {
475     # Parameter einlesen
476     my $file = shift;
477
478     # Existiert die Datei
479     if (-e "$file")
480     {
481       print "\nDie Datei >>$file<< existiert bereits!\n";
482       print "Wird die Datei nicht geloescht, werden die exportierten Daten
              angehaengt!\n";
483       print "Das kann zu Inkompatiblitaeten fuehren!\n";
484       print "Loeschen? (j/n)\n";
485
486       $del_file = ReadLine 0;
487       chop $del_file;
488
489       if ( $del_file eq "j" or $del_file eq "J" )
490       {
491         unlink("$file");
492         print "Datei >>$file<< wurde geloescht\n\n";
493       }
494       else
495       {
496         print "Datei >>$file<< wurde nicht geloescht!\n";
497         print "Das Programm wird weitergefuehrt!\n\n";
498       }
499     }
500   }
```

Quelltext A.1: export.pl

A.2 index.php

```
1   <?php
2
3     # needs list of outsourced information
4     require("inc/extern.inc");
5
6     # destroy all session-data
7     kill_session();
8
9     # start buffer
10    ob_start();
11
12    # start a new session
13    session_start();
14
15    # the site-header
16    require("inc/header.inc");
17
18  ?>
19
20  <div class="main">
21
22    <div class="information">
23      Welcome to <a href="http://www.awi-bremerhaven.de/MET/">MISAWIsta</a> &ndash;
            a prototype of a search interface for AWI's <strong>M</strong>
```

```
        eteorological <strong>I</strong>nformation <strong>S</strong>ystem and
        related publications.  For a more specific search please use the<br />
24    <a href="search_advanced.php"><img src="pics/search.gif" alt="Click here for
        the advanced search" style="vertical-align: middle; width: 18px; height:
        15px; border:0pt" />Advanced Search</a>
25  </div>
26  <div class="indexsearch">
27    <form action="search.php" method="get">
28        <p>
29          <label for="searchfield">Search Field
30            <input type="text" tabindex="0" id="searchfield" name="query" size
                  ="50" alt="search field" value="<?
31            # displays an error-message if nothing was inserted in the field
32            if ( $_GET["em"] == "1" ) echo "please insert valid search term";
                  else echo $_GET["query"]; ?>" />
33          </label>
34          <!--some important variables for the search-->
35          <input type="hidden" name="coord" value="true" />
36          <input type="hidden" name="data" value="1" />
37          <input type="hidden" name="pub" value="1" />
38          <input type="hidden" name="ref" value="i" />
39          <input type="submit" value="Search" class="button" />
40        </p>
41    </form>
42  </div>
43
44 <?php
45
46  # load the RSS-Xml-file
47  $rss = simplexml_load_file("feed/MISAWIsta-news.rss");
48  $attr = $rss->channel->item;
49
50  # the news-box
51  echo "<div class=\"newsbox\">";
52
53    echo "<h5 style=\"margin-top:0px\">Latest News</h5>";
54    echo "<p style=\"font-weight:bold\">".$attr["newsdate"].": ".$attr->title."</
        p>";
55    echo "<p>".$attr->description."</p>";
56
57  echo "</div>";
58
59 ?>
60
61 </div>
62
63 <?php
64
65  # the site-footer
66  require("inc/footer.inc");
67
68  # end of buffer and output of the buffered content
69  ob_end_flush();
70
71 ?>
```

Quelltext A.2: index.php

A.3 search_advanced.php

```
1 <?php
```

```
2
3   # needs list of outsourced information
4   require("inc/extern.inc");
5
6   # destroy all session-data
7   kill_session();
8
9   # start buffer
10  ob_start();
11
12  # start a new session
13  session_start();
14
15  # the site-header
16  require("inc/header.inc");
17
18 ?>
19
20 <div class="main">
21
22   <div class="form">
23     <span class="headline">Free Search Advanced:</span>
24     <form action="search.php" method="get">
25       <p class="title">
26         <label for="searchfield">Search Field
27           <input type="text" tabindex="0" id="searchfield" name="query" size="50"
                  alt="search field" value="<?
28             # displays an error-message if nothing was inserted in the field
29             if ( $_GET["em"] == "1" ) echo "please insert valid search term";
                   else echo $_GET["query"]; ?>" />
30         </label>
31       </p>
32         <div class="coords">
33           <div class="coords-right">
34             <label for="maxLonfields">
35               <input name="maxLon" type="text" id="maxLonfields" size="4"
                     maxlength="4" value="180" /> Max Longitude (East) <br />
36             </label>
37             <label for="minLonfields">
38               <input name="minLon" type="text" id="minLonfields" size="4"
                     maxlength="4" value="-180" /> Min Longitude (West)
39             </label>
40           </div>
41           <div class="coords-left">
42             <label for="maxLatfields">
43               Max Latitude (North) <input name="maxLat" type="text" id="
                     maxLatfields" size="4" maxlength="4" value="90" /><br />
44             </label>
45             <label for="minLatfields">
46               Min Latitude (South) <input name="minLat" type="text" id="
                     minLatfields" size="4" maxlength="4" value="-90" />
47             </label>
48           </div>
49         </div>
50         <div class="checkbox">
51           <label for="datacheckbox1">
52             <input type="checkbox" name="data" id="datacheckbox1" value="1"
                    checked="checked" /> list datasets<br />
53           </label>
54           <label for="pubcheckbox1">
55             <input type="checkbox" name="pub" id="pubcheckbox1" value="1" checked
                    ="checked" /> list publication<br />
56           </label>
```

```
57          </div>
58          <p><input type="submit" value="Search" class="button" /></p>
59        </form>
60      </div>
61
62      <div class="form">
63        <span class="headline">Research Vessels:</span>
64        <form action="search.php" method="get">
65          <p>
66          <label for="expedition">
67            <select name="expedition" id="expedition" size="1">
68              <?  # fills field with results of "expedtion"-function
69                  expedition() ?>
70            </select>
71          </label>
72          <label for="measurement_ship">
73            <select name="measurement" id="measurement_ship" size="1">
74              <?  # fills field with results of "choose_measurement"-function for
                    value "ship"
75                  choose_measurement("ship") ?>
76            </select>
77          </label>
78          </p>
79
80            <div class="checkbox">
81              <label for="datacheckbox2">
82                <input type="checkbox" name="data" id="datacheckbox2" value="1"
                      checked="checked" /> list datasets<br />
83              </label>
84              <label for="pubcheckbox2">
85                <input type="checkbox" name="pub" id="pubcheckbox2" value="1" checked
                      ="checked" /> list publication<br />
86              </label>
87            </div>
88          <input type="hidden" name="coord" value="true" />
89          <p><input type="submit" value="Search" class="button" /></p>
90
91        </form>
92      </div>
93
94      <div class="form">
95        <span class="headline">In situ land based:</span>
96        <form action="search.php" method="get">
97          <p>
98          <label for="platform">
99            <select name="platform" id="platform" size="1">
100             <?  # fills field with results of "platform"-function
101                 platform() ?>
102           </select>
103         </label>
104         <label for="yearcount">
105           <select name="year" id="yearcount" size="1">
106             <?  # fills field with results of "yearcnt"-function for value
                    $stationyear (see in inc/variables.inc)
107                 yearcnt($stationyear) ?>
108           </select>
109         </label>
110         <label for="measurement_platform">
111           <select name="measurement" id="measurement_platform" size="1">
112             <?  # fills field with results of "choose_measurement"-function for
                    value "platform"
113                 choose_measurement("platform") ?>
114           </select>
```

```
115       </label>
116       </p>
117         <div class="checkbox">
118           <label for="datacheckbox3">
119             <input type="checkbox" name="data" id="datacheckbox3" value="1"
                    checked="checked" /> list datasets<br />
120           </label>
121           <label for="pubcheckbox3">
122             <input type="checkbox" name="pub" id="pubcheckbox3" value="1" checked
                    ="checked" /> list publication<br />
123           </label>
124         </div>
125       <input type="hidden" name="coord" value="true" />
126       <p><input type="submit" value="Search" class="button" /></p>
127     </form>
128   </div>
129 <p style="font-size:smaller">* = measurement data will be available in future.
      Meanwhile see below: <a href="http://www.awi-bremerhaven.de/MET">http://www.
      awi-bremerhaven.de/MET</a></p>
130
131 </div>
132
133 <?php
134
135   # the site-footer
136   require("inc/footer.inc");
137
138   # end of buffer and output of the buffered content
139   ob_end_flush();
140
141 ?>
```

Quelltext A.3: search_advanced.php

A.4 help.php

```
1  <?php
2
3    # the site-header
4    require("inc/header.inc");
5
6  ?>
7
8  <div class="main">
9    <div class="help">
10     <span class="helpheadline">Help</span><br />
11     <p class="helpheadline">Search</p>
12
13     <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. For more options
           use the provided Advanced Search.</p>
14
15     <p>MISAWIsta (especially the Pangaea-based part) supports various search
           methods.</p>
16     <ol>
17       <li>The search is <em>not</em> case sensitive<br /><br />
18       Searching for <em>neumayer</em> returns the same results as the term <em>
             NEUMAYER</em> or <em>nEuMaYeR</em>
19       </li>
20       <li>It works with the <em>AND</em> logic by default while searching for
             more keywords<br /><br />
```

```
21        You can use <em>OR</em> if you want either one or another keyword (or both)
              <br />
22        This function is not supported by Fedora (yet)
23        </li>
24        <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 />
25        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 />
26        The &quot;-&quot;-symbol exclude a word from your search, which may not
              contain in the search result<br />
27        The &quot;&sim;&quot;-symbol search for variants in spelling (e.g. <em>&sim
              ;k&ouml;nig</em> finds also <em>karig</em>)<br />
28        This function is not supported by Fedora (yet)
29        </li>
30        <li>Also available are wildcards, which allows to substitute unknown
              characters for part of the item for which you are searching<br /><br />
31        <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 />
32        <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 />
33        <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 />
34        <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 />
35        <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 />
36        <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 />
37        This function is not supported by Fedora (yet)
38        </li>
39        <li>A search term can also contains phrases<br /><br />
40        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 />
41        This function is not supported by Fedora (yet)
42        </li>
43        <li>Pangaea supports the search in specific fields (e.g. <em>author:grobe</
              em> or <em>date:2004</em> and so on) <br /><br />
44        <em>project:</em> search for keywords in projects<br />
45        <em>projectlabel:</em> matches a project label<br />
46        <em>author:</em> search for authors of datasets or assigned references<br
              />
47        <em>citation:author:</em> search for authors of datasets only in the
              citation<br />
48        <em>pi:</em> search for datasets with Principal Investigator (PI)<br />
49        <em>citation:</em> search for keywords in the citation<br />
50        <em>reference:</em> search for keywords in assigned references<br />
51        <em>date:</em> search for datasets or assigned references published in a
              specific year<br />
52        <em>parametername:</em> search for keywords in parameter names<br />
53        <em>methodname:</em> search for keywords in method names<br />
54        <em>eventlabel:</em> search for event labels
55        </li>
56      </ol>
57
58      <p class="helpheadline">Specifics of the Advanced Search</p>
59
```

```
60      <p>The <strong>free search</strong> is equivalent to the simple search, but
            with the option to search in a section of the world. Set the coordinates
            as you want to search for datasets in this area.</p>
61
62      <p>The <strong>search for research vessel-datasets</strong> are easy to use.
            Just choose an entry for each list and klick on <em>Search</em>.</p>
63
64      <p>The <strong>search for in situ land based</strong> is similar to the
            research vessel-search. Also choose the entries in the lists, then klick
            on <em>Search</em>.</p>
65
66      <p class="helpheadline">Dataset-Details</p>
67
68      <p>On detail view of a dataset-result, you can list the fully dataset,
            download a CSV-file (provided by Pangaea) or a XML-file for further
            operations. Just click on the corresponding hyperlink resp. the <em>view
            dataset</em>-button. Please be patient while loading the table, sometimes
             there are many datapoints which need a time to load, depending on your
            connection.</p>
69
70      <p class="helpheadline">FAQ</p>
71
72      <ol class="faq">
73        <li><strong>Question:</strong> I want to view a dataset, which requires a
              login. How is this possible?<br />
74        <strong>Answer:</strong>  In this version of MISAWIsta authentification is
              not possible. You need to go to Pangaea, register to get an account and
               look there for the unpublished datas.</li>
75
76
77      <li><strong>Question:</strong> I want to view a dataset, which requires a
            login. How is this possible?<br />
78      <strong>Answer:</strong>  In this version of MISAWIsta authentification is
            not possible. You need to go to Pangaea, register to get an account and
            look there for the unpublished datas.
79      </li>
80
81      <li><strong>Question:</strong> Where are the tracks of "Heincke" and "Meteor"
            ?<br />
82      <strong>Answer:</strong>  Sorry, but a tracklist for these ships isn't
            reachable at the moment.
83      </li>
84
85      <li><strong>Question:</strong> With and without coordinates, I get the same
            results for publications!<br />
86    <strong>Answer:</strong>  That is right, the publications (belong to Fedora
          System) haven't any coordination metadata. So it isn't possible to search
          with this parameters.
87      </li>
88
89      <li><strong>Question:</strong> The help is so small!<br />
90      <strong>Answer:</strong>  We work on it, please be patient.
91      </li>
92      </ol>
93    </div>
94  </div>
95
96  <?
97
98    # the site-footer
99    require("inc/footer.inc");
100
```

```
101 ?>
```

Quelltext A.4: help.php

A.5 news.php

```
1  <?php
2
3    # needs list of outsourced information
4    require("inc/extern.inc");
5
6    # the site-header
7    require("inc/header.inc");
8
9  ?>
10
11 <div class="main">
12   <div class="links">
13     <span class="linksheadline">MISAWIsta - News</span><br />
14
15 <?php
16
17   # load the RSS-Xml-file
18   $rss = simplexml_load_file("feed/MISAWIsta-news.rss");
19
20   # get the news information
21   foreach($rss->channel->item as $attr)
22   {
23     if ($attr["newsdate"] == $_GET["date"])
24     {
25       $flag = true;
26       $rss_date = $_GET["newsdate"];
27       $rss_title = $attr->title;
28       $rss_description = $attr->description;
29     }
30     else
31     {
32       $flag == false;
33     }
34   }
35
36   # show all news (items)
37   if ( $flag == false )
38   {
39     foreach($rss->channel->item as $attr)
40     {
41       echo "<h4 style=\"margin-bottom:15px; margin-left:35px\">".$attr["newsdate"
             ].": ".$attr->title."</h4>";
42       echo "<p style=\"margin-bottom:40px\">".$attr->description."</p>";
43     }
44   }
45   # show selected item
46   else if ( $flag == true )
47   {
48     echo "<h4 style=\"margin-bottom:15px; margin-left:35px\">".$rss_date.": ".
         $rss_title."</h4>";
49     echo "<p>".$rss_description."</p>";
50   }
51
52 ?>
53
```

```
54   </div>
55 </div>
56
57 <?php
58
59   # the site-footer
60   require("inc/footer.inc");
61
62 ?>
```

Quelltext A.5: help.php

A.6 about.php

```
1  <?php
2
3    # the site-header
4    require("inc/header.inc");
5
6  ?>
7
8  <div class="main">
9    <div class="links">
10     <span class="linksheadline">About</span><br />
11
12     <h5 style="margin-bottom:15px; margin-left:35px">Responsible for this portal
           are:</h5>
13
14     <h4 style="margin-left:35px">Dr. Gert K&ouml;nig-Langlo (meteorologist)</h4>
15     <p style="margin-bottom:25px">Stiftung Alfred-Wegener-Institut für Polar- und
           Meeresforschung<br />
16       in der Helmholtz-Gemeinschaft<br />
17       Bussestrasse 24<br />
18       27570 Bremerhaven<br />
19       Tel.: +49 (0)471 4831-1806<br />
20       Fax: +49 (0)471 4831-1797<br />
21       Email: <a href="mailto:gkoenig@awi-bremerhaven.de">gkoenig@awi-bremerhaven.
             de</a><br />
22       Internet: <a href="http://www.awi-bremerhaven.de/People/show?gkoenig">
             Personal homepage of Dr. Gert K&ouml;nig-Langlo</a><br />
23     </p>
24
25     <h4 style="margin-left:35px">Benny Br&auml;uer (developer)</h4>
26     <p style="margin-bottom:25px">Stiftung Alfred-Wegener-Institut für Polar- und
           Meeresforschung<br />
27       in der Helmholtz-Gemeinschaft<br />
28       Am Handelshafen 12<br />
29       27570 Bremerhaven<br />
30       Tel.: +49 (0)471 4831-1781<br />
31       Fax: +49 (0)471 4831-1590<br />
32       Email: <a href="mailto:bbraeuer@awi-bremerhaven.de">bbraeuer@awi-
             bremerhaven.de</a><br />
33       Internet: <a href="http://www.awi-bremerhaven.de/People/show?bbraeuer">
             Personal homepage of Benny Br&auml;uer</a><br />
34     </p>
35
36     <h4 style="margin-left:35px">Dr. Ana Macario (webmaster)</h4>
37     <p style="margin-bottom:25px">Stiftung Alfred-Wegener-Institut für Polar- und
           Meeresforschung<br />
38       in der Helmholtz-Gemeinschaft<br />
39       Am Handelshafen 12<br />
```

```
40        27570 Bremerhaven<br />
41        Tel.: +49 (0)471 4831-1435<br />
42        Fax: +49 (0)471 4831-1590<br />
43        Email: <a href="mailto:amacario@awi-bremerhaven.de">amacario@awi-
              bremerhaven.de</a><br />
44        Internet: <a href="http://www.awi-bremerhaven.de/People/show?amacario">
              Personal homepage of Dr. Ana Macario</a><br />
45      </p>
46
47      <h5 style="margin-left:35px">More details under:</h5>
48        <p><a href="http://www.awi-bremerhaven.de/AWI/webimpressum.html">http://www
              .awi-bremerhaven.de/AWI/webimpressum.html</a></p>
49
50    </div>
51  </div>
52
53  <?php
54
55    # the site-footer
56    require("inc/footer.inc");
57
58  ?>
```

Quelltext A.6: about.php

A.7 links.php

```
1   <?php
2
3     # the site-header
4     require("inc/header.inc");
5
6   ?>
7   <div class="main">
8     <div class="links">
9       <span class="linksheadline">Links to</span><br />
10      <p class="linksheadline"><a href="http://www.awi-bremerhaven.de/">Alfred
            Wegener Institute for polar and marine research</a></p>
11
12      <p>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>
13
14      <p class="banner"><a href="http://www.awi-bremerhaven.de"><img src="pics/
            AWIlogo.gif" alt="Logo AWI Bremerhaven" /></a></p>
15      <p class="sublinks">Sublinks:</p>
16        <ul>
17          <li><a href="http://www.awi-bremerhaven.de/MET/">Meteorological
                observations made by the AWI</a></li>
18          <li><a href="http://www.awi-bremerhaven.de/MET/rndmetobs.html">
                Meteorological observatories</a></li>
19          <li>
20            <ul>
21              <li><a href="http://www.awi-bremerhaven.de/MET/Neumayer/met.html">
                    Neumayer Station</a></li>
```

```
22            <li><a href="http://www.awi-bremerhaven.de/MET/NyAlesund/index.html">
                  Koldewey Station</a></li>
23            <li><a href="http://www.awi-bremerhaven.de/MET/Polarstern/met.html">
                  Research Vessel Polarstern</a></li>
24          </ul>
25        </li>
26      </ul>
27
28    <p class="linksheadline"><a href="http://www.pangaea.de/" class="headline">
          Pangaea</a></p>
29
30    <p>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>
31
32    <p class="banner"><a href="http://www.pangaea.de" ><img src="pics/PanLogo.gif
          " alt="Logo Pangaea" /></a></p>
33    <p class="sublinks">Sublinks:</p>
34      <ul>
35        <li><a href="http://www.pangaea.de/PangaVista">PangaVista</a></li>
36        <li><a href="http://www.pangaea.de/Info/">Pangaea Info</a></li>
37      </ul>
38
39  </div>
40</div>
41
42<?php
43
44  # the site-footer
45  require("inc/footer.inc");
46
47?>
```

Quelltext A.7: links.php

A.8 search.php

```
1  <?php
2
3    # needs list of outsourced information
4    require("inc/extern.inc");
5
6    # start buffer
7    ob_start();
8
9    // $time_all_start = microtime(true);
10
11   # the site-header
12   require("inc/header.inc");
13
14   # start a new session
15   session_start();
16
17   # ------------------------------------------------------------------
18   # process the variables of the search-request
19   # ------------------------------------------------------------------
20
21     # search-request for ship-data
22     # generate query
23     if ( $_GET["expedition"] )
24     {
```

```
25        $query = $_GET["expedition"];
26        if ( $_GET["measurement"] != null )
27        {
28          $query .= ' citation:"'.$_GET["measurement"].'"';
29        }
30        else
31        {
32          $query .= $_GET["measurement"];
33        }
34      }
35      # search-request for station-data
36      # generate query
37      else if ( $_GET["platform"] )
38      {
39        $query .= $_GET["platform"]." ".$_GET["year"];
40        if ( $_GET["measurement"] != null )
41        {
42          $query .= ' citation:"'.$_GET["measurement"].'"';
43        }
44        else
45        {
46          $query .= $_GET["measurement"];
47        }
48      }
49      # search-request (free search)
50      # generate query
51      else
52      {
53        if (isset($_SESSION["query"]))
54        {
55          $query = $_SESSION["query"];
56        }
57        else
58        {
59          $query = $_GET["query"];
60        }
61      }
62
63      $query = $query." projectlabel:AWI_Meteo";
64
65      # save query in superglobal $_SESSION
66      $_SESSION["query"] = stripslashes($query);
67
68      # give error-message back if no query was inserted in search-field
69      if ( $query == null || $query == "please insert valid search term" )
70      {
71        if ( $_GET["ref"] == "i" )
72        {
73          header("Location: index.php?em=1");
74          exit;
75        }
76        else
77        {
78          header("Location: search_advanced.php?em=1");
79          exit;
80        }
81      }
82
83      # process coordinates
84      if ($_GET["coord"] == true)
85      {
86        $_SESSION["minLat"] = -90;
87        $_SESSION["minLon"] = -180;
```

```
88        $_SESSION["maxLat"] = 90;
89        $_SESSION["maxLon"] = 180;
90      }
91      else
92      {
93        if (!$_SESSION["minLat"]) $_SESSION["minLat"] = $_GET["minLat"];
94        if (!$_SESSION["minLon"]) $_SESSION["minLon"] = $_GET["minLon"];
95        if (!$_SESSION["maxLat"]) $_SESSION["maxLat"] = $_GET["maxLat"];
96        if (!$_SESSION["maxLon"]) $_SESSION["maxLon"] = $_GET["maxLon"];
97      }
98
99      # process offset
100     $offset = $_GET["offset"];
101     if ( $offset == null ) $offset = 0;
102
103     # process data (variable need for listing datasets)
104     if ( $_GET["data"] != null )
105     {
106       unset($_SESSION["data"]);
107       $_SESSION["data"] = $_GET["data"];
108     }
109
110     if (!$_SESSION["data"])
111     {
112       $_SESSION["data"] = $_GET["data"];
113     }
114     if ( $_SESSION["data"] == null ) $_SESSION["data"] = 0;
115
116     # process data (variable need for listing publications)
117     if ($_GET["pub"] != null )
118     {
119       unset($_SESSION["pub"]);
120       $_SESSION["pub"] = $_GET["pub"];
121     }
122
123     if (!$_SESSION["pub"])
124     {
125       $_SESSION["pub"] = $_GET["pub"];
126     }
127     if ( $_SESSION["pub"] == null ) $_SESSION["pub"] = 0;
128
129
130
131   # ------------------------------------------------------------------
132   # process Pangaea-Web Service
133   # ------------------------------------------------------------------
134
135     # process only at the first request or when results exists
136     if ( $_SESSION["tcp"] != 0 || $_SESSION["tcp"] == null )
137     {
138       # generate new object
139       $webservice_pangaea = new webservice_pangaea;
140
141       # handle possible Web Service errors
142       $pangaea_error = $webservice_pangaea->pan_error;
143
144       # on no errors...
145       if ( $pangaea_error != 1 )
146       {
147         # ... check existing session ...
148         if ( $_SESSION["pansession"] == null )
149         {
150           # ... generate new Pangaea-session, if there was no one ...
```

```
151           $_SESSION["pansession"] = $webservice_pangaea->register();
152         }
153         # ... call the Pangaea-Web Service and save the result
154         $result_pangaea = $webservice_pangaea->search_small( $offset, $count );
155       }
156
157       # count the results of pangaea
158       if ( empty($result_pangaea->totalCount) || $result_pangaea->totalCount ==
              null )
159       {
160         $totalcount_pangaea = 0;
161       }
162       else
163       {
164         $totalcount_pangaea = $result_pangaea->totalCount;
165       }
166     }
167     # if no results exist, count is 0
168     else
169     {
170       $totalcount_pangaea = 0;
171     }
172
173   # ------------------------------------------------------------------
174   # process Fedora-Web Service
175   # ------------------------------------------------------------------
176
177     # process only at the first request or when results exists
178     if ( $_SESSION["tcf"] != 0 || $_SESSION["tcf"] == null )
179     {
180       # generate new object
181       $webservice_fedora = new webservice_fedora;
182
183       # handle possible Web Service errors
184       $fedora_error = $webservice_fedora->fed_error;
185
186       # on no errors...
187       if ( $fedora_error != 1 )
188       {
189         if ( $_SESSION["tcf"] == null )
190         {
191           # ... call the Fedora-Web Service and save the result (max 999 results,
                  currently max 100 possible) as an array ...
192           $fedora_array = $webservice_fedora->search_small(999);
193
194           # ... and save this array in superglobal for further handling ...
195           $_SESSION['fedres'] = $fedora_array;
196
197           # count the results of pangaea
198           if ( !empty($fedora_array->resultList) || $fedora_array->resultList !=
                  null )
199           {
200             # "special" count because fedora doesn't return complete list-size (
                    at the moment)
201             foreach( $fedora_array->resultList as $item)
202             {
203               $totalcount_fedora++;
204             }
205           }
206           else
207           {
208             $totalcount_fedora = 0;
209           }
```

```
210         }
211         else
212         {
213           $totalcount_fedora = $_SESSION["tcf"];
214         }
215       }
216     }
217     # if no results exist, count is 0
218     else
219     {
220       $totalcount_fedora = 0;
221     }
222
223     # look for more results
224     if( $totalcount_fedora == 100 && $_SESSION['fed_end'] != 1 )
225     {
226       # get Fedora Session ...
227       $fedsession = $fedora_array->listSession->token;
228
229       # ... for resume the search and save results
230       $fedora_array_resume = $webservice_fedora->resume($fedsession);
231
232       # increment $totalcount_fedora
233       $totalcount_fedora = $totalcount_fedora + count($fedora_array_resume->
              resultList);
234
235       # set superglobals
236       $_SESSION['fedres_resume'] = $fedora_array_resume;
237       $_SESSION['fed_end'] = 1;
238     }
239
240     # make an empty array
241     $result_fedora = array();
242     # use offset (for navigator)
243     $result_fedora_offset = $offset;
244     # calculating for navigator
245     $calc = $result_fedora_offset+$count;
246
247     $_SESSION["fedora_res_end"] = 0;
248
249     if ( $calc > $totalcount_fedora )
250     {
251       $calc = $totalcount_fedora;
252       $result_fedora_offset = $totalcount_fedora - ($totalcount_fedora % $count);
253
254       if ( $offset >= $totalcount_fedora )
255       {
256         $_SESSION["fedora_res_end"] = 1;
257       }
258     }
259
260     # save fedora results between offset and calc in array
261     # this is necessary for a correct function of the navigator and the sidebar
262     for ($i=$result_fedora_offset;$i<$calc;$i++)
263     {
264       # use results of first search
265       if ( $result_fedora_offset < 100 )
266         array_push($result_fedora, $_SESSION['fedres']->resultList[$i]);
267       # use results of resume search
268       else
269         array_push($result_fedora, $_SESSION['fedres_resume']->resultList[$i
              -100]);
270     }
```

```
271
272   # ------------------------------------------------------------------
273   # process the rest
274   # ------------------------------------------------------------------
275
276     # find the max-count
277     if ($totalcount_pangaea >= $totalcount_fedora )
278     {
279       $total_count = $totalcount_pangaea;
280     }
281     else
282     {
283       $total_count = $totalcount_fedora;
284     }
285
286     # save the result counts in superglobals
287     $_SESSION["tcp"] = $totalcount_pangaea;
288     $_SESSION["tcf"] = $totalcount_fedora;
289
290
291   # ------------------------------------------------------------------
292   # generate the result-site with:
293   #   navigator (top)
294   #   sidebar
295   #   results of the pangaea Web Service
296   #   results of the fedora Web Service
297   #   navigator (bootom, identical to the top one)
298   # ------------------------------------------------------------------
299
300     # generate new object
301     $listResult = new listResult;
302
303     # call functions, build the site
304     echo $listResult->navigator( $total_count, $offset, $count );
305     $listResult->sidebar( $offset );
306     $listResult->pangaea_res( $result_pangaea, $pangaea_error, $metadata );
307     $listResult->fedora_res( $result_fedora, $fedora_error );
308     echo $listResult->navigator( $total_count, $offset, $count );
309
310   # the site-footer
311   require("inc/footer.inc");
312
313   # end of buffer and output of the buffered content
314   ob_end_flush();
315
316   # for a chronometry of the script or a part of it, e.g call a Web Service use
        following code at the right place:
317   # to start:
318     # $time_start = microtime(true);
319   # to end:
320     # $time_end = microtime(true);
321   # calculate and display result
322     # $time = $time_end - $time_start;
323     # echo "<p>$time</p>";
324
325 ?>
```

Quelltext A.8: search.php

A.9 showdetails.php

```php
<?php

  # needs list of outsourced information
  require("inc/extern.inc");

  # the site-header
  require("inc/header.inc");

  # doi for pangaea-details
  if ( $_GET["doi"] != null )
  {
    require("inc/details/pangaea.inc");
  }
  # pid for fedora-details
  else if ( $_GET["pid"] != null)
  {
    require("inc/details/fedora.inc");
  }
  else
  {
    echo "No details available!";
  }

  # the site-footer
  require("inc/footer.inc");

?>
```

Quelltext A.9: showdetails.php

A.10 xmlbuilder.php

```php
<?php

  require("inc/extern.inc");

  $doi = $_GET["doi"];

  # generate new object
  $webservice_pangaea = new webservice_pangaea;

  # generate a new session (for example: direct calling of a DOI resp. for
      bookmarking )
  if ( $_SESSION["pansession"] == null || empty($_SESSION["pansession"]) )
  {
    # ... generate new Pangaea-session, if there was no one ...
    $_SESSION["pansession"] = $webservice_pangaea->register();
  }

  # call the Pangaea-Web Service, decode and load it as an SimpleXML-string
  $result_pangaea = simplexml_load_string(base64_decode($webservice_pangaea->
      show_detail($doi)));

  # location of namespace-dfinition
  $xml_str = $result_pangaea->children($metadata);

  #deb($xml_str);

  foreach($xml_str->citation as $cit)
  {
  # string-operations
```

```
28  $doi_download = str_replace("doi:10.1594/PANGAEA.", "", $doi);
29  $doi_tmp  = str_replace("doi:", "", $doi);
30  $doi_filename = str_replace("/", "_", $doi_tmp);
31
32  # open the CSV-file from Pangaea, save in $handle
33  $handle = fopen("http://www.pangaea.de/ddi?datasetid=".$doi_download."&format=
        textfile", "r");
34
35    # start buffer
36    ob_start();
37
38    $flag = false;
39
40    # build XML-string
41    $xmlfile = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\n";
42    $xmlfile .= "<!DOCTYPE MarineDataSet SYSTEM \"http://web.awi-bremerhaven.de//
          php/MISAWIsta/MarineXML/MarineXML_Ver2.0.dtd\">\n";
43    $xmlfile .= "<MarineDataSet creationDate=\"".date("c")."\" name=\"".$doi."\"
          description=\"".$cit->title."\">\n";
44    $xmlfile .= " <Quality>Good</Quality>\n";
45    $xmlfile .= " <Custodian>\n";
46    $xmlfile .= "   <Property name=\"Agency\">Pangaea Information System</
          Property>\n";
47    $xmlfile .= "   <Property name=\"WebSite\">http://www.pangaea.de</Property>\n
          ";
48    $xmlfile .= " </Custodian>\n";
49    $xmlfile .= " <MarineDataRecord ID=\"0\" reject=\"false\">\n";
50
51    $xmlfile .= "   <SpatialReference>\n";
52
53    $xmlfile .= "     <GeoBox>\n";
54    $xmlfile .= "       <Coordinates datum=\"WGS84\">\n";
55    $xmlfile .= "         <Latitude>".number_format($xml_str->extent->geographic
          ->northBoundLatitude, 3, '.', '')."</Latitude>\n";
56    $xmlfile .= "         <Longitude>".number_format($xml_str->extent->geographic
          ->eastBoundLongitude, 3, '.', '')."</Longitude>\n";
57    $xmlfile .= "       </Coordinates>\n";
58    $xmlfile .= "       <Coordinates datum=\"WGS84\">\n";
59    $xmlfile .= "         <Latitude>".number_format($xml_str->extent->geographic
          ->southBoundLatitude, 3, '.', '')."</Latitude>\n";
60    $xmlfile .= "         <Longitude>".number_format($xml_str->extent->geographic
          ->westBoundLongitude, 3, '.', '')."</Longitude>\n";
61    $xmlfile .= "       </Coordinates>\n";
62    $xmlfile .= "     </GeoBox>\n";
63
64    $xmlfile .= "   </SpatialReference>\n";
65
66    $xmlfile .= "   <Source isObservedDate=\"true\" sourceFileName=\"http://www.
          pangaea.de/ddi?datasetid=".$doi_download."&amp;format=textfile\" agency
          =\"Pangaea\" projectID=\"".$doi_download."\">\n";
67    $xmlfile .= "   </Source>\n";
68
69
70
71    # blank array
72    $xml = array();
73
74    # processing $handle
75    while( ($data=fgetcsv($handle, 8192, "\t")) !== false )
76    {
77      // #  search */, delete it and change a flag to true
78      if ($data[0] == "*/")
79      {
```

```
80          unset($data[0]);
81          $flag = true;
82        }
83
84        # insert rest of $handle into the array
85        if ($flag == true)
86        {
87          array_push($xml, $data);
88        }
89      }
90
91      $nr_entries_per_row = count($xml[1]);
92      $nr_entries_per_column = count($xml);
93
94      $xmlfile .= "    <Data numberOfDataObjects=\"".$nr_entries_per_row."\">\n";
95
96      # add other lines as data to XML-string
97      for( $j=0; $j<$nr_entries_per_row; $j++)
98      {
99        $xmlfile .= "      <DataObject index=\"".$j."\" type=\"Primary\" reject=\"
              false\">\n";
100       $xmlfile .= "        <ParameterSet index=\"0\" numberOfParameters=\"0\">\n";
101       $xmlfile .= "          <ValueList numberOfValueSets=\"".(
              $nr_entries_per_column-1)."\">";
102       for( $i=1; $i<$nr_entries_per_column-1; $i++ )
103       {
104         $xmlfile .= $xml[$i][$j].", ";
105       }
106       for( $i=$nr_entries_per_column-1; $i<$nr_entries_per_column; $i++ )
107       {
108         $xmlfile .= $xml[$i][$j];
109       }
110       $xmlfile .= "</ValueList>\n";
111
112       $xmlfile .= "        </ParameterSet>\n";
113       $xmlfile .= "      </DataObject>\n";
114     }
115
116     # close root-element
117
118     $xmlfile .= "    </Data>\n";
119     $xmlfile .= " </MarineDataRecord>\n";
120     $xmlfile .= "</MarineDataSet>\n";
121
122   # close file-handling
123   fclose($handle);
124   }
125   # instructions for browser to let the user download this file:
126   # set filetype
127   header('Content-type: application/xml');
128
129   # set filename
130   header('Content-Disposition: inline');
131   #header('Content-Disposition: attachement; filename="dataset_'.$doi_filename.'.
          xml"');
132
133   # end of buffer and output of the buffered content
134   ob_end_flush();
135
136   # the original
137   echo $xmlfile;
138
```

```
139 ?>
```

Quelltext A.10: xmlbuilder.php

A.11 fedora.inc

```
1  <?php
2
3    # ---------------------------------------------------------------------
4    # process the Fedora details
5    # ---------------------------------------------------------------------
6
7    # start a new session
8    @session_start();
9
10   # get the infos which dataset should be shown
11   $pid = $_GET["pid"];
12
13   #echo " <p class=\"reference\">".$citationdata."</p>";
14   echo "  <p class=\"headline\">".$detailheadline."</p>\n";
15
16   # generate new object
17   $webservice_fedora = new webservice_fedora;
18
19   # call the Fedora-Web Service
20   $result_fedora = $webservice_fedora->show_detail($pid);
21
22   echo "<table class=\"detail\" cellpadding=\"3px\">\n";
23
24   # go through all received datas
25   foreach( $result_fedora->resultList as $item)
26   {
27     # start buffer
28     ob_start();
29
30     if ( $item->title != null )
31     {
32       echo "  <tr>\n"
33           ."    <td class=\"left\">Title:</td>\n"
34           ."    <td>".utf8_decode($item->title)."</td>\n"
35           ."  </tr>\n";
36     }
37
38     echo "  <tr>\n"
39         ."    <td class=\"left\">Archive:</td>\n"
40         ."    <td><a href=\"http://web.awi-bremerhaven.de/php/PKPHarvester/
               viewarchive.php?id=15\">Fedora at AWI</a></td>\n"
41         ."  </tr>\n";
42
43
44     if ( $item->creator != null )
45     {
46       echo "  <tr>\n"
47         ."    <td class=\"left\">Author(s):</td>\n";
48
49       if (is_array($item->creator))
50       {
51         foreach($item->creator as $creator)
52         {
53           $all_authors .= utf8_decode($creator)."; ";
54         }
```

```
55          echo "<td>".rtrim($all_authors,"; ")."</td>\n";
56        }
57        else
58        {
59          echo "<td>".utf8_decode($item->creator)."</td>\n";
60        }
61        echo "  </tr>\n";
62      }
63
64      if ( $item->date != null )
65      {
66        echo "  <tr>\n"
67            ."    <td class=\"left\">Date:</td>\n"
68            ."    <td>".$item->date."</td>\n"
69            ."  </tr>\n";
70      }
71      else
72      {
73        echo "    <td>No date!</td>\n"
74            ."  </tr>\n";
75      }
76
77      if ( $item->subject != null )
78      {
79        echo "  <tr>\n"
80            ."    <td class=\"left\">Subject(s):</td>\n";
81
82        if (is_array($item->subject))
83        {
84          echo "<td>";
85          foreach($item->subject as $subject)
86          {
87            echo utf8_decode($subject)."<br />\n";
88          }
89          echo "</td>\n";
90        }
91        else
92        {
93          echo "<td>".utf8_decode($item->subject)."</td>\n";
94        }
95        echo "  </tr>";
96      }
97
98      if ( $item->description != null )
99      {
100       echo "  <tr>\n"
101         ."    <td class=\"left\">Description:</td>\n";
102
103       if (is_array($item->description))
104       {
105         echo "<td>\n";
106         foreach($item->description as $description)
107         {
108           if ( strpos($description, "doi:") !== false )
109           {
110             echo "<a href=\"http://dx.doi.org/".str_replace("doi:", "",
                      $description)."\">".$description."</a><br />\n";
111           }
112           else
113           {
114             echo "<a href=\"".str_replace("uri:", "", $description)."\">".
                      $description."</a><br />\n";
115           }
```

```
116         }
117         echo "  </td>\n"
118               ."</tr>\n";
119       }
120       else
121       {
122         echo "<td>\n";
123         if ( strpos($item->description, "doi:"))
124         {
125           echo "<a href=\"http://dx.doi.org/".str_replace("doi:", "", $item->
                description)."\">".$item->description."</a>";
126         }
127         else
128         {
129           echo "<a href=\"".str_replace("uri:", "", $item->description)."\">".
                ltrim($item->description, "uri:")."</a>";
130         }
131         echo "  </td>\n"
132               ."</tr>\n";
133       }
134     }
135
136     if ( $item->publisher != null )
137     {
138       echo "  <tr>\n"
139           ."    <td class=\"left\">Publisher:</td>\n"
140           ."    <td>".utf8_decode($item->publisher)."</td>\n"
141           ."  </tr>\n";
142     }
143
144     if ( $item->type != null )
145     {
146       echo "  <tr>\n"
147           ."    <td class=\"left\">Type:</td>\n"
148           ."    <td>".utf8_decode($item->type)."</td>\n"
149           ."  </tr>\n";
150     }
151
152     if ( $item->relation != null )
153     {
154       echo "  <tr>\n"
155           ."    <td class=\"left\">Relation:</td>\n"
156           ."    <td><a href=\"".ltrim($item->relation, "uri:")."\">".ltrim($item
                ->relation, "uri:")."</a></td>\n"
157           ."  </tr>\n";
158     }
159
160     if ( $item->rights != null )
161     {
162       echo "  <tr>\n"
163           ."    <td class=\"left\">Rights:</td>\n"
164           ."    <td><a href=\"".ltrim($item->rights, "uri:")."\">".ltrim($item->
                rights, "uri:")."</a></td>\n"
165           ."  </tr>\n";
166     }
167
168     # search in the description for an abstract
169     if ( !is_array($item->description) && $item->description != null )
170     {
171       $ext_abstract = file_get_contents(ltrim($item->description, "uri:"), "r");
172       preg_match_all("/<!--begin_of_abstract-->(.*?)<!--end_of_abstract-->/ims",
              $ext_abstract, $abstract);
173     }
```

```
174
175     if ( $abstract[0][0] != null )
176     {
177       echo "  <tr>\n"
178           ."    <td class=\"left\">Abstract:</td>\n"
179           ."    <td style=\"text-align:justify\">".$abstract[0][0]."</td>\n"
180           ."  </tr>\n";
181     }
182
183     # end of buffer and output of the buffered content
184     ob_end_flush();
185   }
186   echo "  </table>";
187
188 ?>
```

Quelltext A.11: fedora.inc

A.12 pangaea.inc

```
1  <?php
2
3    # ------------------------------------------------------------------
4    # process the Pangaea details
5    # ------------------------------------------------------------------
6
7    # start a new session
8    @session_start();
9
10   # get the infos which dataset should be shown
11   $doi = $_GET["doi"];
12   #$_SESSION["pancitationdoi"] = $doi;
13   $show_tab = $_GET["show_tab"];
14
15   #echo " <p class=\"reference\">".$citationdata."<br /><a href=\"inc/citation.
         php\">Click here for citation details.</a></p>";
16   echo "  <p class=\"reference\">".$citationdata."<br /><a href=\"#citation\">
         Click here for citation details.</a></p>";
17   echo "  <p class=\"headline\">".$detailheadline."</p>\n";
18
19   # generate new object
20   $webservice_pangaea = new webservice_pangaea;
21
22   # generate a new session (for example: direct calling of a DOI resp. for
         bookmarking )
23   if ( $_SESSION["pansession"] == null || empty($_SESSION["pansession"]) )
24   {
25     # ... generate new Pangaea-session, if there was no one ...
26     $_SESSION["pansession"] = $webservice_pangaea->register();
27   }
28
29   # call the Pangaea-Web Service, decode and load it as an SimpleXML-string
30   $result_pangaea = simplexml_load_string(base64_decode($webservice_pangaea->
         show_detail($doi)));
31
32   # location of namespace-dfinition
33   $xml_str = $result_pangaea->children($metadata);
34
35   # is a login required (unpublished projects)
36   $login = $xml_str->xpath("md:technicalInfo/md:entry[@key='loginRequired']/
         @value");
```

```
37
38  echo "<table class=\"detail\" cellpadding=\"3px\">\n";
39
40  # go through all received datas
41  foreach($xml_str->citation as $cit)
42  {
43    # start buffer
44    ob_start();
45
46    if( $cit->author != null )
47    {
48      echo "  <tr>\n"
49          ."    <td class=\"left\">Title:</td>\n"
50          ."    <td>".utf8_decode($cit->title)."</td>\n"
51          ."  </tr>\n";
52
53      $title=utf8_decode($cit->title);
54      #$_SESSION["pancitationtitle"] = utf8_decode($cit->title);
55      echo "  <tr>\n"
56          ."    <td class=\"left\">Archive:</td>\n"
57          ."    <td><a href=\"http://www.pangaea.de\">Pangaea</a></td>\n"
58          ."  </tr>\n";
59
60      echo "  <tr>\n"
61          ."    <td class=\"left\">Author(s):</td>\n";
62
63      foreach( $cit->author as $author )
64      {
65        $all_authors .= utf8_decode($author->lastName).", ".utf8_decode($author->
              firstName).", ";
66      }
67
68      echo "<td>".rtrim($all_authors,", ")."</td>";
69      echo " </tr>\n";
70      #$_SESSION["pancitationauthors"] = rtrim($all_authors,", ");
71
72      $timestamp = strftime("%Y-%m-%d %H:%M ", strtotime($cit->
            publicationDateTime));
73      #$_SESSION["pancitationyear"] = strftime("%Y", strtotime($cit->
            publicationDateTime));
74
75      echo "  <tr>\n"
76          ."    <td class=\"left\">Date:</td>\n"
77          ."    <td>".$cit->publicationDateTime."</td>\n"
78          ."  </tr>\n";
79    }
80
81    if ( $xml_str->reference != null )
82    {
83      echo "  <tr><td class=\"left\">Reference(s):</td><td></td></tr>\n";
84
85      $ref_nr = 1;
86      foreach($xml_str->reference as $ref)
87      {
88        if( $ref->author != null)
89        {
90          foreach ($ref->author as $ref_author)
91          {
92            $all_ref_authors .= utf8_decode($ref_author->lastName) .", ".
                  utf8_decode($ref_author->firstName). "; ";
93          }
94
```

```
95          echo "    <tr><td class=\"padding\">N° ".arabtoroman($ref_nr)."</td><td
                >".rtrim($all_ref_authors, ", ; ")." in ".$ref->date.": <br />\n";
96          if ($ref->title != null) echo utf8_decode($ref->title)."<br />\n";
97
98          if ($ref->source != null)
99          {
100           echo "<em>".utf8_decode($ref->source);
101           if ( $ref->volume != null ) echo ", volume ".$ref->volume;
102
103           if ( $ref->pages != null )
104           {
105             if ( $ref->pages != "unpublished" )
106             {
107               echo ", pages/medium ".$ref->pages;
108
109             }
110             else
111             {
112               echo ", ".$ref->pages;
113             }
114           }
115           echo "</em><br />\n";
116         }
117
118         if ( $ref->URI != null )
119         {
120           if ( strpos($ref->URI, "doi:") !== false )
121           {
122             echo "<a href=\"http://dx.doi.org/".str_replace("doi:", "", $ref->
                  URI)."\">".$ref->URI."</a>\n";
123           }
124           else
125           {
126             echo "<a href=\"".$ref->URI."\">".$ref->URI."</a>\n";
127           }
128         }
129         echo "    </td>\n"
130             ."  </tr>\n";
131       }
132       $all_ref_authors = null;
133       $ref_nr++;
134     }
135   }
136
137   if( $xml_str->extent->geographic != null )
138   {
139     $extgeo = $xml_str->extent->geographic;
140     echo "  <tr>\n"
141       ."    <td class=\"left\">Spatial Coverage:</td><td></td></tr>\n";
142     if ( $extgeo->northBoundLatitude != null ) echo "<tr><td class=\"padding\">
          North:</td><td>".number_format($extgeo->northBoundLatitude, 3, '.', '')
          ."</td></tr>\n";
143     if ( $extgeo->eastBoundLongitude != null ) echo "<tr><td class=\"padding\">
          East:</td><td>".number_format($extgeo->eastBoundLongitude, 3, '.', '').
          "</td></tr>\n";
144     if ( $extgeo->southBoundLatitude != null ) echo "<tr><td class=\"padding\">
          South:</td><td>".number_format($extgeo->southBoundLatitude, 3, '.', '')
          ."</td></tr>\n";
145     if ( $extgeo->westBoundLongitude != null ) echo "<tr><td class=\"padding\">
          West:</td><td>".number_format($extgeo->westBoundLongitude, 3, '.', '').
          "</td></tr>\n";
146
147     echo "  <tr>\n"
```

```
148         ."    <td class=\"left\">Mean:</td><td></td></tr>\n";
149
150       if ( $extgeo->meanLatitude != null )  echo "<tr><td class=\"padding\">
              Latitude:</td><td>".number_format($extgeo->meanLatitude, 3, '.', '')."
              </td></tr>\n";
151       if ( $extgeo->meanLongitude != null ) echo "<tr><td class=\"padding\">
              Longitude:</td><td>".number_format($extgeo->meanLongitude, 3, '.', '').
              "</td></tr>\n";
152     }
153
154     if( $xml_str->extent->vertical != null )
155     {
156       $extver = $xml_str->extent->vertical;
157       echo "  <tr>\n"
158           ."    <td class=\"left\">Elevation:</td><td></td></tr>\n";
159
160       if ( $extver->minElevation != null )  echo "<tr><td class=\"padding\">min
              :</td><td>".number_format($extver->minElevation, 2, '.', '')." m</td></
              tr>\n";
161       if ( $extver->maxElevation != null )  echo "<tr><td class=\"padding\">max
              :</td><td>".number_format($extver->maxElevation, 2, '.', '')." m</td></
              tr>\n";
162     }
163
164     if( $xml_str->project != null )
165     {
166       echo "  <tr>\n"
167           ."    <td class=\"left\">Project(s):</td>\n"
168           ."    <td>";
169       foreach ( $xml_str->project as $prj )
170       {
171         if ( $prj->projectURL != null )
172         {
173           echo "<a href=\"".$prj->projectURL."\">".utf8_decode($prj->projectName)
                  ."</a> (".utf8_decode($prj->projectLabel).")<br />\n";
174         }
175         else
176         {
177           echo utf8_decode($prj->projectName)." (".utf8_decode($prj->projectLabel
                  ).")<br />\n";
178         }
179       }
180       echo "  </td>"
181           ."</tr>";
182     }
183
184     if( $xml_str->event != null )
185     {
186       echo "  <tr><td class=\"left\">Event(s):</td><td></td></tr>\n";
187
188       $event_count = 0;
189       foreach( $xml_str->event as $evt_tmp )
190       {
191         $event_count++;
192       }
193
194       foreach( $xml_str->event as $eve )
195       {
196         if ( $eve->eventLabel != null )     echo " <tr><td class=\"padding\">
                Label:</td><td> <strong>".utf8_decode($eve->eventLabel)."</strong></
                td></tr>\n";
```

```
197         if ( $eve->optionalLabel != null )  echo " <tr><td class=\"padding\">
                Optional Label:</td><td>".utf8_decode($eve->optionalLabel)."</td></tr
                >\n";
198         if ( $eve->latitude != null )       echo " <tr><td class=\"padding\">
                Latitude:</td><td> ".number_format($eve->latitude, 3, '.', '')."</td
                ></tr>\n";
199         if ( $eve->longitude != null )      echo " <tr><td class=\"padding\">
                Longitude:</td><td> ".number_format($eve->longitude, 3, '.', '')."</
                td></tr>\n";
200         if ( $eve->elevation != null )      echo " <tr><td class=\"padding\">
                Elevation:</td><td> ".number_format($eve->elevation, 2, '.', '')." m
                </td></tr>\n";
201         if ( $eve->datetime != null )       echo " <tr><td class=\"padding\">
                DateTime:</td><td> ".$eve->datetime."</td></tr>\n";
202         if ( $eve->latitude2 != null )      echo " <tr><td class=\"padding\">
                Latitude 2:</td><td> ".number_format($eve->latitude2, 3, '.', '')."</
                td></tr>\n";
203         if ( $eve->longitude2 != null )     echo " <tr><td class=\"padding\">
                Longitude 2:</td><td> ".number_format($eve->longitude2, 3, '.', '')."
                </td></tr>\n";
204         if ( $eve->elevation2 != null )     echo " <tr><td class=\"padding\">
                Elevation 2:</td><td> ".number_format($eve->elevation2, 2, '.', '')."
                 m</td></tr>\n";
205         if ( $eve->datetime2 != null )      echo " <tr><td class=\"padding\">
                DateTime 2:</td><td> ".$eve->datetime2."</td></tr>\n";
206         if ( $eve->location != null )       echo " <tr><td class=\"padding\">
                Location:</td><td> ".utf8_decode($eve->location)."</td></tr>\n";
207         if ( $eve->campaignName != null )   echo " <tr><td class=\"padding\">
                CampaignName:</td><td> ".utf8_decode($eve->campaignName)."</td></tr>\
                n";
208         if ( $eve->campaignUrl != null )    echo " <tr><td class=\"padding\">
                CampaignName:</td><td><a href=\"".$eve->campaignUrl."\">".$eve->
                campaignUrl."</a></td></tr>\n";
209         if ( $eve->basis != null )          echo " <tr><td class=\"padding\">
                Basis:</td><td> ".utf8_decode($eve->basis)."</td></tr>\n";
210         if ( $eve->gearName != null )       echo " <tr><td class=\"padding\">
                GearDevice:</td><td> ".utf8_decode($eve->gearName)."</td></tr>\n";
211         if ( $eve->gearType != null )       echo " <tr><td class=\"padding\">
                GearType:</td><td> ".utf8_decode($eve->gearType)."</td></tr>\n";
212         if ( $eve->gearURL != null )        echo " <tr><td class=\"padding\">
                GearUrl:</td><td><a href=\"".$eve->gearUURL."\">".$eve->gearURL."</a
                ></td></tr>\n";
213         if ( $eve->recovery != null )       echo " <tr><td class=\"padding\">
                Recovery:</td><td> ".utf8_decode($eve->recovery)." m</td></tr>\n";
214         if ( $eve->eventComment != null )   echo " <tr><td class=\"padding\">
                Comment:</td><td> ".utf8_decode($eve->eventComment)."</td></tr>\n";
215
216         if ($event_count > 1)
217         {
218           echo "<tr><td style=\"padding-left:35px\">***</td><td style=\"padding-
                  left:35px\">***</td></tr>";
219         }
220       }
221     }
222
223     if ( $xml_str->dataSetDetails != null )
224     {
225       echo "  <tr>\n"
226         ."    <td class=\"left\">Further Detail(s):</td>\n"
227         ."    <td><a href=\"".$xml_str->dataSetDetails."\">".$xml_str->
                  dataSetDetails."</a></td>\n"
228         ."  </tr>\n";
229     }
```

```
230
231     echo "  <tr>\n"
232         ."    <td class=\"left\">DOI:</td>\n"
233         ."    <td>".$doi."</td>\n"
234         ."  </tr>\n";
235
236     if( $xml_str->size != null )
237     {
238       echo "  <tr>\n"
239         ."    <td class=\"left\">Size:</td>\n"
240         ."    <td>".$xml_str->size."</td>\n"
241         ."  </tr>\n";
242     }
243
244     echo "  <tr>\n"
245         ."    <td class=\"left\"><a style=\"font-weight:bold;text-decoration:none
                ;font-color:black\" id=\"citation\">Citation:</a></td>\n"
246         ."    <td><em>".rtrim($all_authors,"; ")." (".strftime("%Y", strtotime(
                $xml_str->citation->publicationDateTime))."): ".$xml_str->citation->
                title.", PANGAEA, ".$doi."</em></td>\n"
247         ."  </tr>\n";
248
249     echo "  <tr>\n"
250       ."    <td class=\"left\">Parameter(s):</td>\n"
251       ."    </tr>\n";
252     echo"</table>";
253
254     if( $xml_str->geocode != null || $xml_str->dataSeries != null )
255     {
256       echo "<table class=\"parameters\">\n";
257       echo "  <tr>\n"
258         ."    <th>Parameter</th>\n"
259         ."    <th>Short Name</th>\n"
260         ."    <th>Unit</th>\n"
261         ."    <th>Label</th>\n"
262         ."    <th>Principal Investigator</th>\n"
263         ."    <th>Method</th>\n"
264         ."    <th>Comment</th>\n"
265         ."  </tr>\n";
266
267       if( $xml_str->geocode != null )
268       {
269         foreach($xml_str->geocode as $geo)
270         {
271           echo "  <tr>\n";
272           if ( $geo->parameterURL != null )
273           {
274             echo "    <td><a href=\"".$geo->parameterURL."\">".utf8_decode($geo->
                    parameterName)."</a></td>\n";
275           }
276           else
277           {
278             echo "    <td>".utf8_decode($geo->parameterName)."</td>\n";
279           }
280           echo "    <td>".utf8_decode($geo->parameterShortName)."</td>\n"
281               ."    <td style=\"text-align:center\">".utf8_decode($geo->unit)."</
                    td>\n"
282               ."    <td></td>\n"
283               ."    <td></td>\n"
284               ."    <td></td>\n"
285               ."    <td>Geocode</td>\n"
286               ."  </tr>\n";
287         }
```

```
288       }
289
290       if( $xml_str->dataSeries != null )
291       {
292         foreach($xml_str->dataSeries as $data)
293         {
294           $pi = $xml_str->dataSeries->PI;
295           echo "  <tr>\n"
296             ."    <td>".utf8_decode($data->parameterName)."</td>\n"
297             ."    <td>".utf8_decode($data->parameterShortName)."</td>\n"
298             ."    <td style=\"text-align:center\">".utf8_decode($data->unit)."</
                    td>\n"
299             ."    <td>".utf8_decode($data->dataSeriesLabel)."</td>\n";
300
301           if ( $pi->URI != null )
302           {
303             echo " <td style=\"text-align:center\"><a href=\"".$pi->URI."\">".
                      utf8_decode($pi->lastName);
304             if ($pi->firstName != null )
305             {
306               echo ", ".utf8_decode($pi->firstName);
307             }
308             echo "</a></td>\n";
309           }
310           else if( $pi->eMail != null )
311           {
312             echo " <td style=\"text-align:center\"><a href=\"mailto:".$pi->eMail.
                      "\">".utf8_decode($pi->lastName);
313             if ($pi->firstName != null )
314             {
315               echo ", ".utf8_decode($pi->firstName);
316             }
317             echo "</a></td>\n";
318           }
319           else
320           {
321             echo " <td style=\"text-align:center\">".utf8_decode($pi->lastName);
322             if ($pi->firstName != null )
323             {
324               echo ", ".utf8_decode($pi->firstName);
325             }
326             echo "</a></td>\n";
327           }
328
329           echo "    <td>".utf8_decode($data->methodName)."</td>\n"
330               ."    <td>".utf8_decode($data->dataSeriesComment)."</td>\n"
331               ."  </tr>\n";
332         }
333       }
334       echo "  </table>\n";
335     }
336     else
337     {
338       echo "<p style=\"color:red; \">Attention: There are no parameters available
              </p>\n";
339     }
340
341     # end of buffer and output of the buffered content
342     ob_end_flush();
343   }
344
345   # call function dataset() (for the measurement-data of Pangaea)
346   dataset($show_tab, $doi, $xml_str, $login[0]);
```

```
347
348 ?>
```

Quelltext A.12: pangaea.inc

A.13 classes_List.inc

```
1  <?php
2
3    # ------------------------------------------------------------------
4    # class for listing the results
5    # ------------------------------------------------------------------
6
7    class listResult
8    {
9      # turn over the pages
10     function navigator( $total_count, $offset, $count )
11     {
12       # calculate some variables
13       $maxpages = $total_count/$count;
14       $page=$offset/$count;
15
16       if ( $total_count > $count )
17       {
18         # produces the PREV-button
19         echo "<div class=\"navigator\">\n";
20         if ( $page > 0 )
21         {
22           $prev = ($page-1)*$count;
23           if ($prev <= 0) $prev = 0;
24
25           echo "  <a href=\"search.php?offset=".$prev."\">"
26               ."    &lt;&lt; PREV"
27               ."  </a>\n";
28         }
29
30         echo "<span style=\"color:#006ba5\"> | </span>";
31
32         # produces the NEXT-button
33         if ( $page < $maxpages - 1 )
34         {
35           $next = ( $page+1 )*$count;
36
37           echo "  <a href=\"search.php?offset=".$next."\">"
38               ."    NEXT &gt;&gt;"
39               ."  </a>\n";
40         }
41         echo "</div>\n";
42       }
43     }
44
45     # generate the sidebar for further informations and listing options
46     function sidebar( $offset )
47     {
48       echo "<div class=\"sidebar\">";
49
50         # generate a "clean" searchstring
51         $query = str_replace("citation:", "", $_SESSION["query"]);
52         $query = str_replace('"', "", $query);
53         $query = str_replace(' projectlabel:AWI_Meteo', "", $query);
54
```

```
55          echo "<p style=\"margin-top:5px; padding-bottom:5px; border-bottom:0.5px
                solid #ccc\">Query: <strong>".$query."</strong></p>\n";
56
57          echo "<p><strong>".$_SESSION["tcp"]."</strong> datasets found</p>\n"
58              ."<p><strong>".$_SESSION["tcf"]."</strong> publications found</p>\n";
59
60          # save the links in variables for a better handling
61          $d1p0 = "<a href=\"search.php?data=1&amp;pub=0&amp;offset=".$offset."\">
                list datasets</a><br />";
62          $d0p1 = "<a href=\"search.php?data=0&amp;pub=1&amp;offset=".$offset."\">
                list publications</a><br />";
63          $d1p1 = "<a href=\"search.php?data=1&amp;pub=1&amp;offset=".$offset."\">
                list all</a>";
64
65          echo "<div>";
66            if ( $_SESSION["data"] == 1 && $_SESSION["pub"] == 1 )
67            {
68              echo $d1p0;
69              echo $d0p1;
70              echo "<strong>list all</strong><br />";
71            }
72            else if ( $_SESSION["data"] == 0 && $_SESSION["pub"] == 1 )
73            {
74              echo $d1p0;
75              echo "<strong>list publications</strong><br />";
76              echo $d1p1;
77            }
78            else if ( $_SESSION["data"] == 1 && $_SESSION["pub"] == 0 )
79            {
80              echo "<strong>list datasets</strong><br />";
81              echo $d0p1;
82              echo $d1p1;
83            }
84            # if data and pub-variable is 0 get back to search_advanced.php
85            # this can happen if both checkboxes in a advanced search were
                  deselected
86            else
87            {
88              header("Location: search_advanced.php?query=".$_SESSION["query"]."");
89              exit;
90            }
91          echo "<div style=\"margin-top:5px;padding-top:2px;border-top:0.5px solid
                #ccc\">Back to <a href=\"index.php\">Search</a><br />Back to <a href
                =\"search_advanced.php\">Advanced Search</a></div>\n";
92
93          echo "</div>";
94
95        echo "</div>";
96      }
97
98      # list the results of Pangaea
99      function pangaea_res( $result_pangaea, $pangaea_error, $metadata )
100     {
101       # show only when user want it oder simple search was used
102       if ( $_SESSION["data"] == 1)
103       {
104         echo "<div class=\"results\" style=\"margin-bottom:10px\">";
105         echo "<div class=\"resultheadline\">\n"
106             ."Datasets:\n"
107             ."</div>\n";
108
109         if ($_SESSION["tcp"] != 0 && empty($result_pangaea->faultstring) &&
                $result_pangaea->faultstring == null && $pangaea_error != 1)
```

```
110         {
111           foreach($result_pangaea->results as $item)
112           {
113             # load base64-decoded data as SimpleXML-string
114             $xml = simplexml_load_string(base64_decode($item->xml));
115
116             # location of namespaces
117             $xml_str = $xml->children($metadata);
118
119             # format and list all results
120             foreach($xml_str->citation as $cit)
121             {
122               $doi = $cit->URI;
123
124               echo "<div class=\"headline\">";
125               echo "<img src=\"pics/arrow.gif\" alt=\"arrow\" />\n";
126               echo "<a href=\"showdetails.php?doi=".urlencode($doi)."\">";
127
128               if ( $cit->publicationDateTime != null )
129               {
130                 $timestamp = strftime("%Y", strtotime($cit->publicationDateTime))
                      ;
131               }
132               else
133               {
134                 $timestamp = "No publication time";
135               }
136               if ( $cit->author != null )
137               {
138                 echo utf8_decode($cit->author);
139               }
140               else
141               {
142                 echo "No author!";
143               }
144
145               echo " (".$timestamp."): <span class=\"title\">";
146
147               if ( $cit->title != null )
148               {
149                 echo utf8_decode($cit->title)."</span><br />";
150               }
151               else
152               {
153                 echo "No title!"."</span><br />";
154               }
155               echo "</a></div>\n";
156             }
157           }
158         }
159         # error handling
160         else if ( $pangaea_error == 1 )
161         {
162           echo "<p class=\"ferror\">An error has occured &rarr; Could not connect
                   to service!</p>"
163               ."<p class=\"ferror\">Please try again or contact: <a  href=\"
                      mailto:bbraeuer@awi-bremerhaven.de?cc=webmaster@awi-bremerhaven
                      .de&amp;subject=Error in MISAWIsta (Pangaea)\">Benny Bräuer</a
                      ></p>\n";
164         }
165         else if ( !empty($result_pangaea->faultstring) || $result_pangaea->
                faultstring != null )
166         {
```

```
167           echo "<p class=\"ferror\">An error has occured &rarr; $result_pangaea->
                  faultstring</p>"
168           ."<p class=\"ferror\">Please try again or contact: <a  href=\"
                  mailto:bbraeuer@awi-bremerhaven.de?cc=webmaster@awi-bremerhaven
                  .de&amp;subject=Error in MISAWIsta (Pangaea)\">Benny Bräuer</a
                  ></p>\n";
169
170         }
171         else if ( $_SESSION["tcp"] == 0 )
172         {
173           echo "<p class=\"ferror\">No datasets found!<p>\n";
174         }
175         echo "</div>\n";
176       }
177     }
178
179     # list the results of Pangaea
180     function fedora_res( $result_fedora, $fedora_error )
181     {
182       # show only when user want it oder simple search was used
183       if ( $_SESSION["pub"] == 1 )
184       {
185         echo "<div class=\"results\">";
186         echo "<div class=\"resultheadline\">\n"
187             ."Publications:\n"
188             ."</div>\n";
189
190         if ( $_SESSION["tcf"] != 0 && empty($result_fedora->faultstring) &&
                  $result_fedora->faultstring == null && $fedora_error != 1 &&
                  $_SESSION["fedora_res_end"] != 1 )
191         {
192           foreach( $result_fedora as $item)
193           {
194             $pid = $item->pid;
195             echo "<div class=\"headline\">";
196             echo "<img src=\"pics/arrow.gif\" alt=\"arrow\" />\n";
197             echo "<a href=\"showdetails.php?pid=".urlencode($pid)."\">";
198
199               if ( $item->date != null )
200               {
201                 $timestamp = strftime("%Y", strtotime($item->date));
202               }
203               else
204               {
205                 $timestamp = "No publication time";
206               }
207
208               if ( $item->creator != null )
209               {
210                 if (is_array($item->creator))
211                 {
212                   foreach($item->creator as $creator)
213                   {
214                     $all_authors .= utf8_decode($creator)."; ";
215                   }
216                   echo rtrim($all_authors,"; ")."\n";
217                 }
218                 else
219                 {
220                   echo utf8_decode($item->creator);
221                 }
222                 $all_authors = null;
223               }
```

```
224               else
225               {
226                 echo "No author!";
227               }
228
229               echo " (".$timestamp."): <span class=\"title\">";
230
231               if ( $item->title != null )
232               {
233                 echo utf8_decode($item->title)."</span><br />";
234               }
235               else
236               {
237                 echo "No title!"."</span><br />";
238               }
239
240             echo "</a></div>";
241           }
242         }
243         # error handling
244         else if ( $fedora_error == 1 )
245         {
246           echo "<p class=\"ferror\">An error has occured &rarr; Could not connect
                   to service!</p>"
247               ."<p class=\"ferror\">Please try again or contact: <a  href=\"
                     mailto:bbraeuer@awi-bremerhaven.de?cc=webmaster@awi-bremerhaven
                     .de&amp;subject=Error in MISAWIsta (Fedora)\">Benny Bräuer</a
                     ></p>\n";
248         }
249         else if ( !empty($result_fedora->faultstring) || $result_fedora->
                faultstring != null )
250         {
251           echo "<p class=\"ferror\">An error has occured &rarr; $result_fedora->
                faultstring</p>"
252               ."<p class=\"ferror\">Please try again (e.g. click on \"NEXT\") or
                     contact: <a  href=\"mailto:bbraeuer@awi-bremerhaven.de?cc=
                     webmaster@awi-bremerhaven.de&amp;subject=Error in MISAWIsta (
                     Fedora)\">Benny Bräuer</a></p>\n";
253         }
254         else if ( $_SESSION["tcf"] == 0 )
255         {
256           echo "<p class=\"ferror\">No datasets found!<p>\n";
257         }
258         else if ( $_SESSION["fedora_res_end"] == 1 )
259         {
260           echo "<p class=\"ferror\">No more datasets found!<p>\n";
261         }
262         echo "</div>";
263       }
264     }
265   }
266
267
268 ?>
```

Quelltext A.13: classes_List.inc

A.14 classes_WS.inc

```
1 <?php
2
```

```
3   # ---------------------------------------------------------------------
4   # classes for Web Service-handling
5   # ---------------------------------------------------------------------
6
7   # the uber-class
8   class webservice
9   {
10    var $wsdl;
11    var $client;
12    var $error;
13  }
14
15  # class for Pangaea
16  class webservice_pangaea extends webservice
17  {
18    var $session;
19
20    # creates new SOAP-client
21    function webservice_pangaea()
22    {
23      try
24      {
25        # are the servers reachable?
26        $fp = @fsockopen("www.pangaea.de", 80, $errno, $errstr, 5);
27        $fp2 = @fsockopen("ws.pangaea.de", 80, $errno, $errstr, 5);
28
29        if (!$fp || !$fp2)
30        {
31          $this->error = 1;
32          return $this->error;
33        }
34        else
35        {
36          # the location of the WSDL
37          $this->wsdl = "http://ws.pangaea.de/ws/services/PangaVista?wsdl";
38
39          # generate the SOAP-client
40          $this->client = new SoapClient($this->wsdl, array("trace" => 1, "
                exceptions" => 1, "connection_timeout" => 10));
41
42          # check the client ...
43          $function = $this->client->__getFunctions();
44
45          # ... return error on fail
46          if ( empty($function) ||  connection_status() > 0)
47          {
48            $this->error = 1;
49            return $this->error;
50          }
51        }
52        fclose($fp);
53        fclose($fp2);
54      }
55      catch (SoapFault $fault)
56      {
57        $this->error = 1;
58        return $this->error;
59      }
60    }
61
62    # register a Pangaea-session
63    function register()
64    {
```

```
65      try
66      {
67        $this->session = $this->client->registerSession(null, $_SERVER['
              HTTP_USER_AGENT'], $_SERVER['REMOTE_HOST'], "PangaVista");
68        if (is_soap_fault($this->session) || connection_status() > 0 )
69        {
70          $this->error = 1;
71          return $this->error;
72        }
73
74        # return the session
75        $session = $this->session;
76        return $session;
77      }
78      catch (SoapFault $fault)
79      {
80        return $fault;
81      }
82    }
83
84    # the search in Pangaea
85    function search_small( $offset, $count )
86    {
87        try
88        {
89          $result_pangaea = $this->client->search($_SESSION["pansession"], iconv(
                "ISO-8859-1","UTF-8",$_SESSION["query"]), $_SESSION["minLat"],
                $_SESSION["minLon"], $_SESSION["maxLat"], $_SESSION["maxLon"],
                $offset, $count);
90
91          if (is_soap_fault($result_pangaea) || connection_status() > 0 )
92          {
93            $this->error = 1;
94            return $this->error;
95          }
96
97          # return the result
98          return $result_pangaea;
99        }
100       catch (SoapFault $fault)
101       {
102         return $fault;
103       }
104   }
105
106   # show the details of a DOI-entry
107   function show_detail( $doi )
108   {
109     try
110     {
111       $result_pangaea = $this->client->metadata($_SESSION["pansession"], $doi);
112
113       # return result
114       return $result_pangaea;
115     }
116     catch (SoapFault $fault)
117     {
118       return $fault;
119     }
120   }
121 }
122
123 # class for Fedora
```

```
124 class webservice_fedora extends webservice
125 {
126   var $resultfields;
127
128   # creates new SOAP-client
129   function webservice_fedora()
130   {
131     try
132     {
133       # is the server reachable?
134       $fp = @fsockopen("web.awi-bremerhaven.de", 8080, $errno, $errstr, 5);
135
136       if (!$fp)
137       {
138         $this->error = 1;
139         return $this->error;
140       }
141       else
142       {
143         # the location of the WSDL
144         $this->wsdl = "http://web.awi-bremerhaven.de:8080/fedora/access/soap?
              wsdl";
145
146         # generate the SOAP-client
147         $this->client = new SoapClient($this->wsdl, array("trace" => 1, "
              exceptions" => 1, "connection_timeout" => 10));
148
149         # which fields in Fedora System the Web Service should search for
150         $this->resultfields = array("title",
151                                     "creator",
152                                     "subject",
153                                     "description",
154                                     "publisher",
155                                     "contributor",
156                                     "date",
157                                     "type",
158                                     "format",
159                                     "identifier",
160                                     "source",
161                                     "language",
162                                     "relation",
163                                     "coverage",
164                                     "rights",
165                                     "pid"
166                                   );
167
168         # return the fieldlist
169         return $resultfields;
170       }
171       fclose($fp);
172     }
173     catch (SoapFault $fault)
174     {
175       $this->error = 1;
176       return $this->error;
177     }
178   }
179
180   # the search in Fedora
181   function search_small( $count )
182   {
183     $fed_tmp_query = $_SESSION["query"];
184
```

```
185       # format the query
186       $fed_tmp_query = str_replace("citation:", "", $fed_tmp_query);
187       $fed_tmp_query = str_replace('"', "", $fed_tmp_query);
188       $fed_tmp_query = str_replace(' projectlabel:AWI_Meteo', "", $fed_tmp_query)
              ;
189
190       # generate Fedora-query
191       $fedora_query = array("conditions" => null, "terms" => "*".utf8_encode(
              $fed_tmp_query)."*");
192
193       try
194       {
195         $result_fedora = $this->client->findObjects($this->resultfields, $count,
                $fedora_query);
196
197         # return the result
198         return $result_fedora;
199       }
200       catch ( SoapFault $fault)
201       {
202         return $fault;
203       }
204     }
205
206     # resume the search
207     function resume( $fid )
208     {
209       try
210       {
211         $next_result_fedora = $this->client->resumeFindObjects($fid);
212
213         # return the result
214         return $next_result_fedora;
215       }
216       catch ( SoapFault $fault)
217       {
218         return $fault;
219       }
220     }
221
222     # show the details
223     function show_detail( $pid )
224     {
225       try
226       {
227         $fedora_query = array("conditions" => null, "terms" => $pid."*");
228         $result_fedora = $this->client->findObjects($this->resultfields, 1,
                $fedora_query);
229
230         # return the result
231         return $result_fedora;
232       }
233       catch( SoapFault $fault)
234       {
235         return $fault;
236       }
237     }
238   }
239
240 ?>
```

Quelltext A.14: classes_WS.inc

A.15 extern.inc

```php
<?php

  # ------------------------------------------------------------------
  # collecting point for extern files (includes)
  # special php.ini-entries
  # ------------------------------------------------------------------

  # some variables
  require("variables.inc");

  # the classes
  require("classes_List.inc");
  require("classes_WS.inc");

  # the functions
  require("functions.inc");

  # some ini-changes, see http://www.php.net/ for more details
  ini_set("zend.ze1_compatibility_mode","0");
  ini_set("default_socket_timeout","15");

?>
```

Quelltext A.15: extern.inc

A.16 footer.inc

```
1
2     <div class="footer">
3       <h5 style="margin:0px auto 5px auto">Navigate to:</h5>
4       <a href="index.php">Search</a> |
5       <a href="search_advanced.php">Advanced search</a> |
6       <a href="news.php">News</a> |
7       <a href="links.php">Links</a> |
8       <a href="help.php">Help</a> |
9       <a href="about.php">About</a>
10    </div>
11
12    <div class="footer">
13      <a href="http://validator.w3.org/check?uri=referer">
14        <img src="pics/xhtml.png" style="border-style:none" alt="Valid XHTML 1.1!
              " /></a>
15      <a href="http://jigsaw.w3.org/css-validator/check/referer/" style="margin-
            left:1em">
16        <img src="pics/css.png" style="border-style:none" alt="Valid CSS!" /></a>
17      <a href="http://www.contentquality.com/mynewtester/cynthia.exe?Url1=http:%2
            F%2Fweb.awi-bremerhaven.de%2Fphp%2FMISAWIsta%2Findex.php&amp;rptmode=2"
              style="margin-left:1em">
18        <img src="pics/wai.png" style="border-style:none" alt="WAI Level AA!"
              /></a>
19      <a href="http://web.awi-bremerhaven.de/php/MISAWIsta/feed/MISAWIsta-news.
            rss" style="margin-left:1em">
20        <img src="pics/rss.png" style="border-style:none" alt="RSS News Feed!"
              /></a>
21    </div>
22
23    <div class="footer">
```

```
24        <a href="mailto:bbraeuer@awi-bremerhaven.de">bbraeuer@awi-bremerhaven.de</a
              >
25      </div>
26
27    </body>
28  </html>
```

Quelltext A.16: footer.inc

A.17 functions.inc

```
1  <?php
2
3  # creates ship-infolist with Polarstern-track-data form MisawiDB
4  function expedition()
5  {
6    # connect to db
7    $dbproc = sybase_connect('AWI', 'www_Misawi', 'WWW_mISAWI') or $error = 1;
8    echo "<option value=\"Polarstern\">All tracks of Polarstern (AWI)</option>\n";
9    if ( $error != 1 )
10   {
11     $db_query = "select   Reise,
12                           Fahrtabschnitt,
13                           Abfahrtshafen,
14                           Ankunftshafen
15                 from      Reisen
16                 where     Reise != \"Werftprobefahrt\"
17                 group by  Reise
18                 order by  Reise";
19
20     # save result
21     $db_res = sybase_query($db_query, $dbproc);
22
23     # print all tracks into list
24     while($row = sybase_fetch_object($db_res))
25     {
26         $reise = str_replace(" ", "-", $row->Reise);
27         echo "  <option value=\"citation:".$reise."/".$row->Fahrtabschnitt."\">&
               rarr; "
28             . $row->Reise."/".$row->Fahrtabschnitt.": "
29             . $row->Abfahrtshafen." - ".$row->Ankunftshafen."</option>\n";
30     }
31
32     # close connection
33     sybase_close($dbproc);
34   }
35   else
36   {
37     echo "<option value=\"\">&rarr; Could not connect to Track-Database of
           Polarstern</option>";
38   }
39
40   # other ships
41   echo "<option value=\"Heincke\">All tracks of Heincke (AWI/BAH)</option>";
42   echo "<option value=\"Meteor\">All tracks of Meteor (BMBF/DFG)</option>";
43 }
44
45 # creates station-list
46 function platform()
47 {
48   # array for stations
```

```
49    $platform_awi = array
50    (
51      "Neumayer",
52      "Koldewey"
53    );
54
55    # print all stations
56    foreach($platform_awi as $val)
57    {
58      echo "<option value=\"".$val."\">".$val." (AWI)</option>\n";
59    }
60  }
61
62  # creates year-list (actual year - 1)
63  function yearcnt($year)
64  {
65    $now=strftime("%Y", time());
66    for($year; $year<$now; $year++)
67    {
68      echo "<option value=\"citation:".$year."\">".$year."</option>\n";
69    }
70    echo "<option value=\"\">All years</option>\n";
71  }
72
73  # creates measurement-list
74  function choose_measurement($type)
75  {
76    # for ships
77    if ( $type == "ship" )
78    {
79        echo "<option value=\"Meteorological observations\">3-hourly routine
              synoptic observations</option>\n";
80        echo "<option value=\"upper air soundings\">Upper air soundings *</option>\
              n";
81        echo "<option value=\"POLDAT-data\">POLDAT-data, averaged over 10 minutes
              *</option>\n";
82    }
83
84    # for stations
85    if ( $type == "platform" )
86    {
87        # attention: for station 3-hourly routine synoptic observations pangaea is
              called "Meteorological synoptical observations" instead of "
              Meteorological observations"
88        echo "<option value=\"Meteorological synoptical observations\">3-hourly
              routine synoptic observations</option>\n";
89        echo "<option value=\"upper air soundings\">Upper air soundings *</option>\
              n";
90        echo "<option value=\"Surface radiation\">Surface radiation and mast
              measurements *</option>\n";
91    }
92    echo "<option value=\"\">All kind of measurement</option>\n";
93  }
94
95  # convert arabic numbers to roman
96  function arabtoroman($val)
97  {
98    $roman   = array( "M","CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "
          IV", "I");
99    $arabian = array(1000, 900, 500,  400, 100,   90,  50,  40,   10,    9,   5,
            4,   1);
100   $val  = IntVal($val);
101
```

```
102   for ($i=0; $i<count($arabian); $i++)
103   {
104     while ($val >= $arabian[$i])
105     {
106        $number .= $roman[$i];
107        $val -= $arabian[$i];
108     }
109   }
110   return $number;
111 }
112
113 # open / close dataset-table
114 function dataset($show_tab, $doi, $xml_str, $login)
115 {
116   # ... only shown when no login is need
117   if ( $login == "false" )
118   {
119     $size = $xml_str->size;
120     $doi_download = str_replace("doi:10.1594/PANGAEA.", "", $doi);
121
122     echo "<div style=\"text-align:left\">\n";
123
124     # create link for XML-file
125     echo "<br /><a href=\"xmlbuilder.php?doi=".urlencode($doi)."\">Download
          dataset as MarineXML-file</a> (in development, for testing purpose only)"
          ;
126     # create link for CSV-file
127     echo "<br /><a href=\"http://www.pangaea.de/ddi?datasetid=".$doi_download."&
          amp;format=textfile\">Download dataset as tab-delimited textfile from
          Pangaea.de</a>\n";
128
129     if ( $show_tab == null or $show_tab == 0 )
130     {
131       $inf_string = str_replace(" data points","", $size);
132
133       # the show-button
134       echo "<div>";
135       echo "      <form action=\"showdetails.php\" method=\"get\">\n"
136           ."        <p>\n"
137           ."        <input type=\"hidden\" name=\"doi\" value=\"".$doi."\" />\n"
138           ."        <input type=\"hidden\" name=\"show_tab\" value=\"1\" />\n"
139           ."        <input type=\"submit\" value=\"view dataset\" class=\"button
                \" />\n"
140           ."        </p>\n"
141           ."      </form>\n";
142       echo "</div>";
143
144       # a hint for size of the dataset-table
145       echo "<p>\n";
146       if ( $inf_string > 5000 and $inf_string <= 20000 )
147       {
148         echo " <b>Attention: Large size (".$size.") - maybe long loading time!</b
              >";
149       }
150       else if ($inf_string > 20000)
151       {
152         echo " <span style=\"color:red;font-weight:bold\">Attention: Very large
              size (".$size.") - maybe long loading time!</span>";
153       }
154       echo "</p>\n";
155
156     }
157     else if ( $show_tab == 1 )
```

```
158     {
159       # a close-button
160       echo "<div>";
161       echo "       <form action=\"showdetails.php\" method=\"get\">\n"
162           ."         <p>\n"
163           ."         <input type=\"hidden\" name=\"doi\" value=\"".$doi."\" />\n"
164           ."         <input type=\"hidden\" name=\"show_tab\" value=\"0\" />\n"
165           ."         <input type=\"submit\" value=\"close dataset\" class=\"button
                \" />\n"
166           ."         </p>\n"
167           ."       </form>\n";
168       echo "</div>";
169
170       echo "<div style=\"margin-bottom:40px\">\n"
171           ."</div>";
172     }
173
174     if ( $show_tab == 1 )
175     {
176       # open CSV-filehandling
177       $handle = fopen("http://www.pangaea.de/ddi?datasetid=".$doi_download."&
              format=textfile", "r");
178
179         # start buffer
180         ob_start();
181
182         $flag = false;
183
184         echo "<table class=\"dataset\">";
185
186         # blank array
187         $table = array();
188
189         # processing $handle
190         while( ($data=fgetcsv($handle, 4096, "\t")) !== false )
191         {
192           # search */, delete it and change a flag to true
193           if ($data[0] == "*/")
194           {
195             unset($data[0]);
196             $flag = true;
197           }
198
199           # insert rest of $handle into the array
200           if ($flag == true)
201           {
202             array_push($table, $data);
203           }
204         }
205
206         $nr_field = count($table[1]);
207
208         # bold headline
209         echo "<tr>";
210         for( $i=0; $i<$nr_field; $i++ )
211           {
212             echo "<td style=\"text-align:center;white-space:nowrap\"><b> ".
                  $table[1][$i]." </b></td>\n";
213           }
214         echo "</tr>";
215
216         $nr_entries = count($table);
217
```

```
218         # the data
219         for( $i=2; $i<$nr_entries; $i++ )
220         {
221           echo"<tr>";
222           for( $j=0; $j<$nr_field; $j++)
223           {
224             echo "<td style=\"text-align:center;white-space:nowrap\"> ".
                  $table[$i][$j]." </td>\n";
225           }
226           echo "</tr>";
227         }
228         echo "</table>";
229 
230       # end of buffer and output of the buffered content
231       ob_end_flush();
232 
233       # close file-handling
234       fclose($handle);
235 
236       # another close-button
237       echo "<div>";
238       echo "       <form action=\"showdetails.php\" method=\"get\">\n"
239           ."         <p>\n"
240           ."         <input type=\"hidden\" name=\"doi\" value=\"".$doi."\" />\n"
241           ."         <input type=\"hidden\" name=\"show_tab\" value=\"0\" />\n"
242           ."         <input type=\"submit\" value=\"close dataset\" class=\"button
                \" />\n"
243           ."         </p>\n"
244           ."       </form>\n";
245       echo "</div>";
246     }
247     echo "</div>\n";
248   }
249   else
250   {
251     echo "<div style=\"margin:25px; color:red; font-weight:bold;\">Login required
          ! Please use <a href=\"http://www.pangaea.de\">Pangaea</a> to register if
           you are/were a member of this project and receive further details.
          Otherwise ask the <a href=\"http://pangaea.de/Info/\">responsibles of the
           Pangaea Group</a>.</div>";
252   }
253 }
254 
255 # formatted dump of a variable
256 function deb($var)
257 {
258   echo "<pre>\n";
259   print_r($var);
260   echo "</pre>\n";
261 }
262 
263 # really kills a session
264 function kill_session()
265 {
266   # clear cookie
267   unset($_COOKIE[session_name()]);
268   # clear superglobal $_SESSION and its content
269   unset($_SESSION);
270   # destroy session data and prevent warnings
271   @session_destroy();
272 }
273 
```

```
274 ?>
```

Quelltext A.17: functions.inc

A.18 header.inc

```
1
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
      DTD/xhtml11.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
4 <head>
5   <title>MISAWIsta - Prototype: Search for meteorology publications and datasets
        </title>
6
7   <?  # load extra CSS-information for IE
8       $browser = $_SERVER["HTTP_USER_AGENT"];
9       if (strpos($browser, "IE"))
10      {
11        echo '<link rel="stylesheet" href="styles/for_ie.css" type="text/css" />'
              ;
12      }
13      else
14      {
15        echo '<link rel="stylesheet" href="styles/for_others.css" type="text/css"
               />';
16      }
17  ?>
18
19  <link rel="stylesheet" href="styles/styles.css" type="text/css" />
20  <link rel="schema.DC" href="http://purl.org/dc/elements/1.1/" />
21  <link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />
22  <link rel="alternate" title="Latest MISAWIsta News" href="http://web.awi-
        bremerhaven.de/php/MetVista/feed/MISAWIsta-news.rss" type="application/rss+
        xml" />
23  <link rel="alternate" title="Latest Pangaea Datasets" href="http://www.pangaea.
        de/PangaVista/latest-datasets.rss" type="application/rss+xml" />
24  <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
25  <meta http-equiv="content.language" content="en" />
26  <meta name="author" content="Benny Bräuer" />
27  <meta name="robots" content="index,nofollow" />
28  <meta name="DC.creator" content="Benny Bräuer" />
29  <meta name="DC.title" content="MISAWIsta - Prototype" />
30  <meta name="DC.subject" content="AWI Search for meteorology publications and
        datasets" />
31  <meta name="DC.description" content="Search for meteorology publications and
        datasets using Web Service Technology. Whole portal acts as a SOAP-client."
         />
32  <meta name="DC.publisher" content="Alfred Wegener Institute for polar and
        marine research" />
33  <meta name="DC.contributor" content="Dr. Ana Macario" />
34  <meta name="DC.identifier" scheme="DCTERMS.URI" content="http://web.awi-
        bremerhaven.de/php/MISAWIsta/index.php" />
35  <meta name="DC.source" content="publications and datasets" />
36  <meta name="DC.relation" content="diploma thesis" />
37  <meta name="DC.rights" content="Copyright Benny Bräuer, all rights reserved" />
38  <meta name="DC.date" scheme="DCTERMS.W3CDTF" content="2005-06-11T12:00:00+01:00
        " />
39  <meta name="DC.type" scheme="DCTERMS.DCMIType" content="Collection" />
40  <meta name="DC.coverage" scheme="DCTERMS.TGN" content="Bremerhaven" />
41  <meta name="DC.format" scheme="DCTERMS.IMT" content="text/html" />
42  <meta name="DC.language" scheme="DCTERMS.RFC3066" content="en" />
```

```
43
44 </head>
45
46 <body>
47   <div class="banner">
48     <a href="http://www.awi-bremerhaven.de"><img src="pics/banner.jpg" alt="
         banner" /></a>
49   </div>
50
51   <div class="header">
52     <h5 style="margin:0px auto 5px auto">Navigate to:</h5>
53     <a href="index.php">Search</a> |
54     <a href="search_advanced.php">Advanced search</a> |
55     <a href="news.php">News</a> |
56     <a href="links.php">Links</a> |
57     <a href="help.php">Help</a> |
58     <a href="about.php">About</a>
59   </div>
60
61   <!--<p style="font-size:larger; font-weight:bold; color:red; text-align:center"
       >BETA-Version, development is still in progress!</p>-->
```

Quelltext A.18: header.inc

A.19 variables.inc

```
1 <?php
2
3   # ------------------------------------------------------------------
4   # place for some fix variables
5   # ------------------------------------------------------------------
6
7   # namespace-URI
8   $metadata = 'http://www.pangaea.de/MetaData';
9
10  # year of the first station-measurement
11  $stationyear = 1981;
12
13  # nr. of results per resultlist
14  $count = 10;
15
16  $citationdata = "Always give the proper citation when using this data!";
17  $detailheadline = "Record Details";
18
19 ?>
```

Quelltext A.19: variables.inc

A.20 MISAWIsta-news.rss

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <rss version="2.0">
3   <channel>
4
5     <title>MISAWIsta News</title>
6     <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php</link>
7     <description>Newspage of MISAWIsta</description>
8     <language>en-en</language>
```

```
9     <copyright>2005 AWI Bremerhaven</copyright>
10    <pubDate>Wed, 24 Aug 2005 13:00:00 +0100</pubDate>
11    <image>
12      <url>http://web.awi-bremerhaven.de/php/MISAWIsta/pics/banner.jpg</url>
13      <title>MISAWI Banner</title>
14      <link>http://www.awi-bremerhaven.de</link>
15    </image>
16
17    <item newsdate="2005-09-02">
18      <title>Filter implemented</title>
19      <description>By today, MISAWIsta will only find meteorology datasets of Dr.
            Gert K&amp;ouml;nig-Langlo to guarantee the primary sense of the
          portal. More datasets will follow in the future.</description>
20      <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php?date=2005-09-02<
          /link>
21    </item>
22
23    <item newsdate="2005-09-01">
24      <title>XML changes to MarineXML</title>
25      <description>For a standardized XML-format to contain the metadata, the
          australian MarineXML-format is used by now. See &lt;a href="http://www.
          marinexml.net/"&gt;http://www.marinexml.net&lt;/a&gt; and &lt;a href="
          http://www.metoc.gov.au/products/prod/marinexml.html"&gt;http://www.
          metoc.gov.au/products/prod/marinexml.html&lt;/a&gt; for further details
          .</description>
26      <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php?date=2005-09-01<
          /link>
27    </item>
28
29    <item newsdate="2005-08-19">
30      <title>RSS-feed available</title>
31      <description>Today, a RSS-feed for the news was implemented. Now it is
          possible to get the latest news for MISAWIsta with a RSS-Reader (or
          applications which support RSS, like &lt;a href="http://www.mozilla.org
          /products/thunderbird/"&gt;Mozilla Thunderbird&lt;/a&gt;).</description
          >
32      <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php?date=2005-08-19<
          /link>
33    </item>
34
35    <item newsdate="2005-08-05">
36      <title>MISAWIsta goes public</title>
37      <description>MISAWIsta is no longer a beta-version. I'm proud to present a
          release candidate of this prototype. Hope, it will help you a little
          bit to find one's way in the world of meteorology data.</description>
38      <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php?date=2005-08-05<
          /link>
39    </item>
40
41    <item newsdate="2005-06-08">
42      <title>MISAWIsta development startet</title>
43      <description>With the development of MISAWIsta the main part of my diploma
          thesis was started.</description>
44      <link>http://web.awi-bremerhaven.de/php/MISAWIsta/news.php?date=2005-06-08<
          /link>
45    </item>
46
47  </channel>
48 </rss>
```

Quelltext A.20: MISAWIsta-news.rss

A.21 styles.css

```
a {
  text-decoration: underline;
}

a:link {
  color: #006ba5;
  text-decoration:none;
}

a:active {
  color: #006ba5;
}

a:visited {
  color: #006ba5;
  text-decoration: none;
}

a:hover {
  color: #006ba5;
  text-decoration: underline;
}

input {
  font-size: 12px;
}

textarea {
  font-size: 12px;
}

select {
  font-size: 12px;
}

hr {
  color: #666;
  background-color: #666;
  height: 1px;
  border: 0;
}

p {
  margin:5px auto;
}


div.main {
  width:693px;
  margin-left: auto;
  margin-right: auto;
  background: #FFF;
  overflow: auto;
}

div.sidebar {
  width:180px;
  float:right;
  background: #ebebeb;
  border: 1px #666 dotted;
```

```
61    padding: 10px;
62    text-align:right;
63  }
64
65  div.left {
66    float: left;
67    margin-right: 15px;
68  }
69
70  div.right {
71    float: right;
72    margin-left: 15px;
73  }
74
75  div.spacer {
76    font-size: 2px;
77    clear: both;
78  }
79
80  input.button {
81    font-weight: bold;
82    color: #FFF;
83    background: #006ba5;
84    border-top: 1px #CCC solid;
85    border-left: 1px #999 solid;
86    border-bottom: 1px #333 solid;
87    border-right: 1px #000 solid;
88    padding-left: 10px;
89    padding-right: 10px;
90    padding-top: 2px;
91    padding-bottom: 2px;
92    margin-top:10px;
93  }
94
95  input.button:hover {
96    border-top: 1px #333 solid;
97    border-left: 1px #000 solid;
98    border-bottom: 1px #CCC solid;
99    border-right: 1px #999 solid;
100 }
101
102 input.details {
103   font-weight: bold;
104   margin-top: 5px;
105   background-color: #006ba5;
106   color:#FFF;
107   border:0.5px solid #DDDDDD;
108   border-top: 1px #CCC solid;
109   border-left: 1px #999 solid;
110   border-bottom: 1px #333 solid;
111   border-right: 1px #000 solid;
112   padding-left: 10px;
113   padding-right: 10px;
114   padding-top: 2px;
115   padding-bottom: 2px;
116 }
117
118 input.details:hover {
119   border-top: 1px #333 solid;
120   border-left: 1px #000 solid;
121   border-bottom: 1px #CCC solid;
122   border-right: 1px #999 solid;
123 }
```

```
124
125 div.banner {
126   text-align:center;
127   margin-bottom: 10px;
128 }
129
130 div.banner img{
131   border:none;
132 }
133
134 div.information {
135   text-align:center;
136   font-size:larger;
137 }
138
139 div.navigator {
140   width:200px;
141   text-align:center;
142   font-weight:bold;
143   font-size:larger;
144   background:#ebebeb;
145   border:1px dotted;
146   margin:10px auto;
147   padding:10px;
148 }
149
150 div.newsbox {
151   width:400px;
152   text-align:center;
153   background:#ebebeb;
154   margin:10px auto;
155   padding:5px;
156 }
157
158 div.form {
159   border: 1px #ddd solid;
160   background: #ebebeb;
161   padding: 10px;
162   margin-bottom:10px;
163   text-align:center;
164 }
165
166 div.form form {
167   clear:both;
168 }
169
170 div.indexsearch {
171   padding: 10px;
172   margin-bottom:10px;
173   text-align:center;
174 }
175
176 div.indexsearch span.headline {
177   margin-bottom:10px;
178   float:left;
179   font-weight:bold;
180 }
181
182 div.form span.headline {
183   margin-bottom:10px;
184   float:left;
185   font-weight:bold;
186 }
```

```
187
188 div.form div.checkbox {
189   margin:10px auto 2px;
190   width:20%;
191   text-align:left;
192 }
193
194 div.form div.coords {
195   border:1px solid #eee;
196   margin:10px auto;
197   width:460px;
198   text-align:center;
199 }
200
201 div.form div.coords-left {
202   margin-top:4px;
203   margin-left:0px;
204   width:200px;
205   text-align:right;
206 }
207
208 div.form div.coords-right {
209   margin-top:4px;
210   margin-left:0px;
211   width:200px;
212   text-align:left;
213   float:right;
214 }
215
216 div.results {
217   text-align: left;
218   padding: 15px 15px 15px 5px;
219 }
220
221 div.resultheadline {
222   font-style:italic;
223   font-size:larger;
224   padding-left: 10px;
225   margin-bottom: 10px;
226   color: #006ba5;
227 }
228
229 div.headline {
230   color: #006ba5;
231   font-size: 14px;
232   font-weight: bold;
233   margin-left: 25px;
234   margin-bottom: 10px;
235 }
236
237 div.headline span.title {
238   font-weight: normal;
239   color: #000;
240 }
241
242 div.header {
243   text-align: center;
244   margin-bottom: 20px;
245 }
246
247 div.footer {
248   text-align: center;
249   padding-top: 15px;
```

```
250   clear:both;
251 }
252
253 div.links{
254   text-align:justify;
255 }
256
257 div.links img{
258   border:0px;
259 }
260
261 div.links span.linksheadline{
262   margin-left: 25px;
263   margin-bottom: 125px;
264   font-weight: bold;
265   font-size: larger;
266 }
267
268 div.links p {
269   margin-left: 55px;
270   margin-right: 55px;
271 }
272
273 div.links p.linksheadline {
274   margin-top: 25px;
275   font-variant: small-caps;
276   font-size: medium;
277   font-weight:bold
278 }
279
280 div.links p.banner {
281   margin-bottom:25px;
282   float:right;
283 }
284
285 div.links p.sublinks {
286   margin-left: 85px;
287   margin-top: 20px;
288   margin-bottom: -10px;
289   font-weight:bold;
290 }
291
292 div.links ul {
293   margin-bottom: 40px;
294 }
295
296 div.links li {
297   margin-left: 45px;
298   list-style-position:inside;
299   list-style-type:none;
300 }
301
302 div.links div.faq {
303   margin-left: 15px;
304   margin-bottom: 25px;
305
306 }
307
308 div.help span.helpheadline{
309   margin-left: 25px;
310   margin-bottom: 125px;
311   font-weight: bold;
312   font-size: larger;
```

```
313 }
314
315 div.help p {
316   margin-left: 55px;
317   margin-right: 55px;
318 }
319
320 div.help p.helpheadline {
321   margin-top: 25px;
322   font-variant: small-caps;
323   font-size: medium;
324   font-weight:bold
325 }
326
327 div.help ol, div.help ol.faq {
328   margin-bottom: 40px;
329 }
330
331 div.help li {
332   margin-left: 45px;
333   margin-bottom: 15px;
334   list-style-position: outside;
335   list-style-type: upper-roman;
336   background-color:#eee;
337 }
338
339 div.help ol.faq li
340 {
341   margin-left: 45px;
342   margin-bottom: 15px;
343   list-style-position: outside;
344   list-style-type: none;
345   background-color:#eee;
346 }
347
348 div.help div.faq {
349   margin-left: 15px;
350   margin-bottom: 25px;
351 }
352
353 table.detail {
354   width: 95%;
355   text-align:left;
356 }
357
358 table.detail tr {
359   margin-bottom:120px;
360 }
361
362 table.detail td.left {
363   width: 150px;
364   font-weight:bold;
365   vertical-align:top;
366 }
367
368 table.detail td.padding {
369   width: 150px;
370   padding-left:20px;
371   vertical-align:top;
372 }
373
374 table.parameters {
375   width: 95%;
```

```
376   border:1px solid #ddd;
377   border-collapse:separate;
378   empty-cells:show;
379   text-align:left;
380 }
381
382 table.parameters td{
383   border:1px solid #ddd;
384 }
385
386 table.parameters th{
387   font-weight:bold;
388   text-align:center;
389   border:1px solid #ddd;
390 }
391
392 table.dataset {
393   border:1px solid #ddd;
394   border-collapse:separate;
395   empty-cells:show;
396 }
397
398 table.dataset td {
399   border:1px solid #ddd;
400 }
401
402 p.reference {
403   color: red;
404   font-weight: bold;
405   text-align:right;
406 }
407
408 p.headline
409 {
410   font-size:200%;
411   font-variant: small-caps;
412   text-align:left;
413 }
414
415 p.ferror {
416   font-weight:bold;
417   color:red;
418   padding-left:55px;
419 }
```

Quelltext A.21: styles.css

A.22 for_ie.css

```
1 html, body {
2   font-family: Verdana,Arial,Geneva,Helvetica,sans-serif;
3   font-size: 10pt;
4   background: #FFFFFF;
5   text-align:center;
6 }
```

Quelltext A.22: for_ie.css

A.23 for_others.css

```
html, body {
  font-family: Verdana,Arial,Geneva,Helvetica,sans-serif;
  font-size: 10pt;
  background: #FFFFFF;
}
```

Quelltext A.23: for_others.css

A.24 MetaData.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Michael
    Diepenbroek (MARUM) -->
<xs:schema targetNamespace="http://www.pangaea.de/MetaData" elementFormDefault="
    qualified" attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org
    /2001/XMLSchema" xmlns:md="http://www.pangaea.de/MetaData">
  <xs:complexType name="MetaDataType">
    <xs:sequence>
      <xs:element name="citation" type="md:DataSetCitationType"/>
      <xs:element name="size" type="xs:string"/>
      <xs:element name="extent" type="md:ExtentType" minOccurs="0"/>
      <xs:element name="reference" type="md:ReferenceType" minOccurs="0"
          maxOccurs="unbounded"/>
      <xs:element name="project" type="md:ProjectType" minOccurs="0" maxOccurs="
          unbounded"/>
      <xs:element name="dataSetComment" type="xs:string" minOccurs="0"/>
      <xs:element name="dataSetDetails" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="event" type="md:EventType" minOccurs="0" maxOccurs="
          unbounded"/>
      <xs:element name="geocode" type="md:ParameterType" minOccurs="0" maxOccurs=
          "unbounded"/>
      <xs:element name="dataSeries" type="md:DataSeriesType" minOccurs="0"
          maxOccurs="unbounded"/>
      <xs:element name="keywords" type="md:KeywordsType" minOccurs="0"/>
      <xs:element name="technicalInfo" type="md:TechnicalInfoType" minOccurs="0"/
          >
    </xs:sequence>
  </xs:complexType>
  <xs:element name="MetaData" type="md:MetaDataType">
    <xs:annotation>
      <xs:documentation>root element for MetaData</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="DataSetCitationType">
    <xs:complexContent>
      <xs:extension base="md:CitationType">
        <xs:sequence>
          <xs:element name="publicationDateTime" type="xs:dateTime" minOccurs="0"
              />
          <xs:element name="parentURI" type="xs:anyURI" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="CitationType">
    <xs:sequence>
      <xs:element name="author" type="md:ResponsiblePartyType" maxOccurs="
          unbounded"/>
```

```
38        <xs:element name="date" type="xs:gYear"/>
39        <xs:element name="title" type="xs:string"/>
40        <xs:element name="source" type="xs:string" minOccurs="0"/>
41        <xs:element name="URI" type="xs:anyURI" minOccurs="0"/>
42      </xs:sequence>
43      <xs:attribute name="sequenceNo" type="xs:int" use="optional" default="0"/>
44      <xs:attribute name="id" type="xs:ID" use="optional"/>
45    </xs:complexType>
46    <xs:complexType name="ExtentType">
47      <xs:sequence>
48        <xs:element name="geographic" minOccurs="0">
49          <xs:complexType>
50            <xs:sequence>
51              <xs:element name="westBoundLongitude" type="xs:double"/>
52              <xs:element name="eastBoundLongitude" type="xs:double"/>
53              <xs:element name="southBoundLatitude" type="xs:double"/>
54              <xs:element name="northBoundLatitude" type="xs:double"/>
55              <xs:element name="meanLongitude" type="xs:double"/>
56              <xs:element name="meanLatitude" type="xs:double"/>
57              <xs:element name="location" type="xs:string" minOccurs="0"/>
58            </xs:sequence>
59          </xs:complexType>
60        </xs:element>
61        <xs:element name="temporal" minOccurs="0">
62          <xs:complexType>
63            <xs:sequence>
64              <xs:element name="minAge" type="xs:double" minOccurs="0"/>
65              <xs:element name="maxAge" type="xs:double" minOccurs="0"/>
66              <xs:element name="minDateTime" type="xs:dateTime" minOccurs="0"/>
67              <xs:element name="maxDateTime" type="xs:dateTime" minOccurs="0"/>
68            </xs:sequence>
69          </xs:complexType>
70        </xs:element>
71        <xs:element name="vertical" minOccurs="0">
72          <xs:complexType>
73            <xs:sequence>
74              <xs:element name="minElevation" type="xs:double" minOccurs="0"/>
75              <xs:element name="maxElevation" type="xs:double" minOccurs="0"/>
76            </xs:sequence>
77          </xs:complexType>
78        </xs:element>
79      </xs:sequence>
80    </xs:complexType>
81    <xs:complexType name="DataSeriesType">
82      <xs:complexContent>
83        <xs:extension base="md:ParameterType">
84          <xs:sequence>
85            <xs:element name="PI" type="md:ResponsiblePartyType" minOccurs="0"/>
86            <xs:element name="dataSeriesLabel" type="xs:string" minOccurs="0"/>
87            <xs:element name="dataSeriesComment" type="xs:string" minOccurs="0"/>
88            <xs:element name="methodName" type="xs:string" minOccurs="0"/>
89            <xs:element name="methodURL" type="xs:anyURI" minOccurs="0"/>
90          </xs:sequence>
91        </xs:extension>
92      </xs:complexContent>
93    </xs:complexType>
94    <xs:complexType name="ParameterType">
95      <xs:sequence>
96        <xs:element name="parameterName" type="xs:string"/>
97        <xs:element name="parameterShortName" type="xs:string"/>
98        <xs:element name="unit" type="xs:string" minOccurs="0"/>
99        <xs:element name="parameterGroup" type="xs:string" minOccurs="0"/>
100       <xs:element name="parameterURL" type="xs:anyURI" minOccurs="0"/>
```

```
101     </xs:sequence>
102     <xs:attribute name="id" type="xs:ID" use="optional"/>
103   </xs:complexType>
104   <xs:complexType name="EventType">
105     <xs:sequence>
106       <xs:element name="eventLabel" type="xs:string"/>
107       <xs:element name="optionalLabel" type="xs:string" minOccurs="0"/>
108       <xs:element name="latitude" type="xs:double" minOccurs="0"/>
109       <xs:element name="longitude" type="xs:double" minOccurs="0"/>
110       <xs:element name="elevation" type="xs:double" minOccurs="0"/>
111       <xs:element name="datetime" type="xs:dateTime" minOccurs="0"/>
112       <xs:element name="latitude2" type="xs:double" minOccurs="0"/>
113       <xs:element name="longitude2" type="xs:double" minOccurs="0"/>
114       <xs:element name="elevation2" type="xs:double" minOccurs="0"/>
115       <xs:element name="datetime2" type="xs:dateTime" minOccurs="0"/>
116       <xs:element name="location" type="xs:string" minOccurs="0"/>
117       <xs:element name="eventComment" type="xs:string" minOccurs="0"/>
118       <xs:element name="campaignName" type="xs:string" minOccurs="0"/>
119       <xs:element name="campaignURL" type="xs:anyURI" minOccurs="0"/>
120       <xs:element name="basis" type="xs:string" minOccurs="0"/>
121       <xs:element name="gearName" type="xs:string" minOccurs="0"/>
122       <xs:element name="gearURL" type="xs:anyURI" minOccurs="0"/>
123       <xs:element name="gearType" type="xs:string" minOccurs="0"/>
124       <xs:element name="recovery" type="xs:double" minOccurs="0"/>
125     </xs:sequence>
126     <xs:attribute name="id" type="xs:ID" use="optional"/>
127   </xs:complexType>
128   <xs:complexType name="ProjectType">
129     <xs:sequence>
130       <xs:element name="projectName" type="xs:string"/>
131       <xs:element name="projectLabel" type="xs:string" minOccurs="0"/>
132       <xs:element name="projectURL" type="xs:anyURI" minOccurs="0"/>
133     </xs:sequence>
134     <xs:attribute name="id" type="xs:ID" use="optional"/>
135   </xs:complexType>
136   <xs:complexType name="ReferenceType">
137     <xs:complexContent>
138       <xs:extension base="md:CitationType">
139         <xs:sequence>
140           <xs:element name="volume" type="xs:string" minOccurs="0"/>
141           <xs:element name="pages" type="xs:string" minOccurs="0"/>
142         </xs:sequence>
143       </xs:extension>
144     </xs:complexContent>
145   </xs:complexType>
146   <xs:complexType name="ResponsiblePartyType">
147     <xs:sequence>
148       <xs:element name="lastName" type="xs:string"/>
149       <xs:element name="firstName" type="xs:string" minOccurs="0"/>
150       <xs:element name="eMail" type="xs:anyURI" minOccurs="0"/>
151       <xs:element name="URI" type="xs:anyURI" minOccurs="0"/>
152     </xs:sequence>
153     <xs:attribute name="sequenceNo" type="xs:int" use="optional"/>
154     <xs:attribute name="id" type="xs:ID" use="optional"/>
155   </xs:complexType>
156   <xs:complexType name="KeywordsType">
157     <xs:sequence>
158       <xs:element name="techKeyword" type="xs:string" minOccurs="0" maxOccurs="
            unbounded"/>
159       <xs:element name="keyword" type="xs:string" minOccurs="0" maxOccurs="
            unbounded"/>
160     </xs:sequence>
161   </xs:complexType>
```

```
162   <xs:complexType name="TechnicalInfoType">
163     <xs:sequence>
164       <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
165         <xs:complexType>
166           <xs:attribute name="key" type="xs:string" use="required"/>
167           <xs:attribute name="value" type="xs:anySimpleType" use="required"/>
168         </xs:complexType>
169       </xs:element>
170     </xs:sequence>
171   </xs:complexType>
172   <xs:complexType name="SearchResultType">
173     <xs:sequence>
174       <xs:element name="citation" type="md:ShortCitationType"/>
175       <xs:element name="reference" type="md:ShortCitationType" minOccurs="0"
              maxOccurs="unbounded"/>
176       <xs:element name="size" type="xs:string"/>
177       <xs:element name="extent" type="md:ExtentType"/>
178       <xs:element name="technicalInfo" type="md:TechnicalInfoType" minOccurs="0"/
              >
179     </xs:sequence>
180   </xs:complexType>
181   <xs:element name="SearchResult" type="md:SearchResultType">
182     <xs:annotation>
183       <xs:documentation>root element for SearchResult</xs:documentation>
184     </xs:annotation>
185   </xs:element>
186   <xs:complexType name="ShortCitationType">
187     <xs:sequence>
188       <xs:element name="author" type="xs:string"/>
189       <xs:element name="date" type="xs:gYear"/>
190       <xs:element name="title" type="xs:string"/>
191       <xs:element name="source" type="xs:string" minOccurs="0"/>
192       <xs:element name="URI" type="xs:anyURI" minOccurs="0"/>
193       <xs:element name="publicationDateTime" type="xs:dateTime" minOccurs="0"/>
194       <xs:element name="parentURI" type="xs:anyURI" minOccurs="0"/>
195       <xs:element name="volume" type="xs:string" minOccurs="0"/>
196       <xs:element name="pages" type="xs:string" minOccurs="0"/>
197     </xs:sequence>
198     <xs:attribute name="sequenceNo" type="xs:int" use="optional" default="0"/>
199   </xs:complexType>
200 </xs:schema>
```

Quelltext A.24: MetaData.xsd

B Eidesstattliche 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.

Midlum, den 29. September 2005 _____