



## **reUSE**

### **digital master files of printed publications**

#### **Deliverable D1.4 Digital Repository - Germany**

**Date of delivery: 4.July.2005**

**Author's organisation: Computer and Media Service**

**Version: 1.0**

## Executive Summary

The Deliverable 1.4, Digital Repositories in Germany, is divided into three main chapters:

- Chapter 1 – edoc-Server documentation
- Chapter 2 – MetaIn
- Chapter 3 – MetaOut/MetaSearch

Chapter 1 is called “edoc-Server documentation” and includes an overview about the server-structure and -functionality. It describes the used hard- and software, the workflow, agreements, used standards and the Metadata handling.

Chapter 2 describes the “MetaIn” PHP-frontend. It is the input-interface of the edoc-Server and is available via webpage <https://edoc.hu-berlin.de/cgi/dokupload/dokupload.cgi>.

Chapter 3 is the description of the PHP-frontend “MetaOut/MetaSearch” that is used as an output-interface.



## **Chapter 1**

### **edoc-Server description**

#### **Deliverable D1.4 Digital Repository – Germany**

# Index

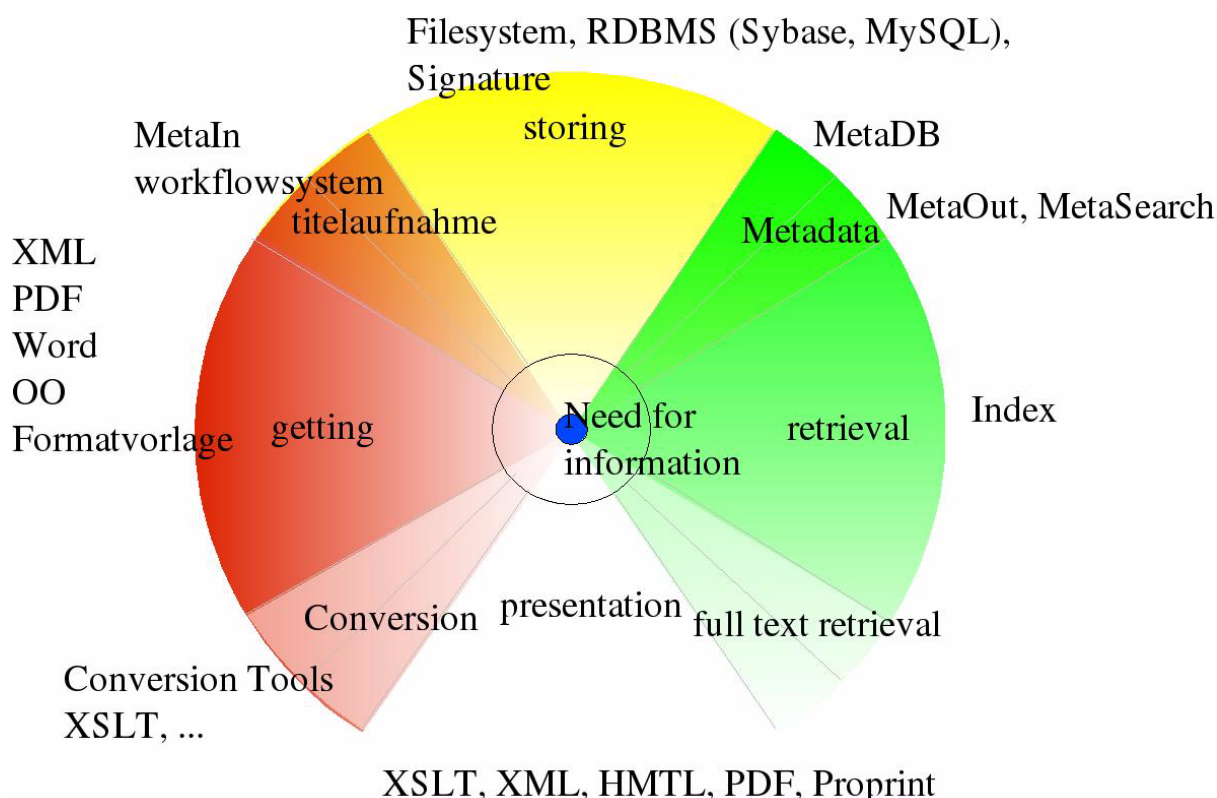
1	Goals.....	4
1.1	Mission.....	4
1.2	The Collection Mandate of Humboldt University Library for Digital Documents....	5
1.2.1	Copyrights .....	5
1.2.3	Author agreement .....	5
1.2.3	Creative Common licenses.....	5
1.3	Digital Documents.....	5
1.4	Technical Characteristics of the Document and Publication Server .....	6
1.5	Organisational Regulations .....	6
1.6	The "Electronic Publishing" Group.....	7
2	Hard- and Software .....	7
3	Edoc mapping against OAIS .....	7
3.1	Ingest.....	8
3.1.1	Receive Submission .....	9
3.1.2	Quality Assurance .....	10
3.1.3	Generate AIP .....	10
3.1.4	Generate Descriptive Information .....	10
3.1.5	Coordinate Updates .....	10
3.2	Archival Storage.....	11
3.2.1	Receive Data .....	11
3.2.2	Manage Storage Hierarchy .....	11
3.2.3	Replace Media.....	11
3.2.4	Error Checking .....	11
3.2.5	Disaster Recovery .....	12
3.2.6	Backup-media.....	12
3.2.7	Provide Data .....	12
3.3	Data Management .....	13
3.3.1	Administer Database .....	13
3.3.2	Perform Queries .....	13
3.3.3	Generate Report.....	13
3.3.4	Receive Database Updates .....	13
3.4	Administration.....	14
3.4.1	Negotiate Submission.....	14
3.4.2	Manage System Configuration.....	14
3.4.3	Archival Information Update .....	14
3.4.4	Physical Access Control.....	14
3.4.5	Establish Standards and Policies .....	15
3.4.6	Audit Submission .....	15
3.4.7	Activate Requests .....	15
3.4.8	Customer Service .....	15
3.5	Preservation Planning.....	16
3.5.1	Monitor Designated Community.....	16
3.5.2	Monitor Technology.....	16
3.5.3	Develop Preservation Strategies and Standards .....	16
3.5.4	Develop Packaging Designs and Migration Plans .....	16
3.6	Access.....	17
3.6.1	Co-ordinate Access Activities .....	17
3.6.2	Generate DIP .....	17
3.6.3	Deliver Response.....	17

4	Meta-DB and Metadata .....	17
4.1	Metadata Database .....	18
4.1.1	Database Structure.....	18
4.1.2	Storing and Finding Data .....	19
4.2	Metadata .....	20
4.2.1	Dublin Core .....	20
4.2.2	ProPrint Metadata.....	20
4.2.3	MetaDis .....	20
4.2.4	OAI-Interface .....	20

# 1 Goals

In spite of the fact that the final specification of a complete digital publishing framework is still unknown due to fast development and lack of proven standards, the Document and Publication Server of Humboldt University Berlin represents a working system and the experiences collected during the time of employment can be used as a starting point for a new generation repositories covering the long term preservation as well the information retrieval issues.

Graphic 1: Sketch of Parts of edoc-Repository



## 1.1 Mission

### Objectives and Criteria for Content of the Document and Publication Server of Humboldt University

The Document and Publication Server offers the organisational and technical framework to all members of Humboldt University for publishing scientific documents digitally. In course of this joint services offered by the Computer and Media Services and the University Library scientific documents of high importance are being published on the internet under strict quality control.

The digital documents are provided with persistent identifiers and addresses, and are indexed within national and international library catalogues, search engines, and other reference tools. The document and publication server provides protection against distortion by using digital signatures and digital time stamps. Furthermore, long-time preservation of digital publications is guaranteed.

Operation and development of the document and publication server are integrated into national and international initiatives and projects like the »Networked Digital Library of Theses and Dissertations« (NDLTD) or the »Open Archives Initiative« (OAI)

## 1.2 The Collection Mandate of Humboldt University Library for Digital Documents

The collection mandate of Humboldt University Library consists of collecting, cataloguing, and archiving all the scientific documents published by the members of Humboldt University. It refers to digitally born documents as well as digital versions of printed documents.

Also included are significant historic documents from the University Library and other institutions that are digitised due to terms of content, conservatory aspects, or the requirements of place-independent use.

Intellectual property rights will be preserved. Publishing the document with the publication and document server does not prevent publication elsewhere, like scientific journals or other document servers. According to the recommendations of the German Science Council all scientists of Humboldt University are asked to secure further rights of use when negotiating with publishers. Later, maybe after a certain qualifying period, they will be asked to also publish their documents with the Humboldt University Document Server.

### 1.2.1 Copyrights

Observation of copyrights and rights of use for third parties are solely with the authors or the editors of the digital documents.

### 1.2.3 Author agreement<sup>1</sup>

The author submit the non-exclusive publication rights to the university library. So the university library could offer a worldwide internet access to the publication.

The authors are still the owner of their copyrights and it's possible to offer rights for other utilisations.

The agreement defines a submission publication format.

### 1.2.3 Creative Common licenses

Offering Creative Common licenses is still in process. The decision to offer CC licenses was made but the organisational and technical conditions must be created.

There are a few problems to solve. For example there isn't legal certainty about the question "What is the exact meaning of 'transforming'?". Is it transforming if you change the content of a document? For sure! But is it transforming if you only convert/transform a document from one format into another?

## 1.3 Digital Documents

The term »digital document« as used here is defined as a document based on text and images, that is stored in digital form on a data medium, and that is distributed via computer networks. In the future this term will be extended toward multimedia documents that include audio and video sequences.

**A document to be published with the document and publication server should meet the following requirements:**

1. It is to be distributed for the public.
2. It is not a dynamic document. With any changes in the document a new version will be

---

<sup>1</sup> [http://edoc.hu-berlin.de/e\\_autoren/erklaerung.php](http://edoc.hu-berlin.de/e_autoren/erklaerung.php)

stored.

3. It fulfils the technical parameters given by the Computer and Media Services and the University Library of Humboldt University.

**The following categories of digital documents will be stored and distributed:**

1. Single publications and publication series of Humboldt University with scientific content, as the public lectures of Humboldt University or the publication series of the University Library,
2. Single publications and publication series published by staff of Humboldt University, i.e. collections, conference papers, research reports, journals (e-journals),
3. Documents that are mandatory to be published according to examination rules and regulations (theses and dissertations),
4. Single publications and publication series of institutions and persons associated with Humboldt University,
5. Documents of Humboldt University students like master theses and seminar papers, should those be recommended by a lecturer.

Distribution of these documents can be restricted to local or temporary use only.

## **1.4 Technical Characteristics of the Document and Publication Server**

1. By providing and proving qualified digital signatures the digital documents receive a legally valid certificate of authenticity. The assignment of these qualified digital signature occur in full accordance with the German Digital Signatures Law (Signaturgesetz - SigG).
2. The digital documents are provided with individual and persistent addresses. This enables direct access to the document.
3. Retrieving the digital documents is possible via library catalogues, queries of the bibliographical metadata, search terms within the structures of the digital documents, via alphabetical and indexing systems, and via dynamically produced lists and indexes.
4. For indexing, storage and archiving of the digital documents international standards like the guidelines of the Open Archives Initiative (OAI) are employed and developed.
5. When using SGML/XML an archiving time span of 50 years is guaranteed. Preserving other formats depends on the availability of these formats, their viewer software, and their conversion tools.

## **1.5 Organisational Regulations**

1. The document and publication server of Humboldt University is a joint operation of the Computer and Media Services and the University Library.
2. The electronic publication is free of charge for Humboldt University staff and members of associated institutions.
3. Submitting the digital documents for distribution via the Humboldt University document server takes place in the University Library.
4. Additional services that are necessary for publication, like processing digital documents or conversion into other formats, are conducted by staff of the Computer and Media Services or the University Library.<sup>5</sup> Services that need significant additional work will be charged according to the scale of charges of the Computer and Media Services and the University Library.



## 1.6 The "Electronic Publishing" Group

The joint working group of the University Library and the Computer and Media Services aims towards the systematic improvement of the electronic services at Humboldt University. The main focus is to broaden the usage and to propagate the possibilities of electronic publishing.

**Therefore we focus on the following tasks.**

- The acquisition and handling of digital publications of Humboldt University according to international standards and using new technologies.
- The realisation of a worldwide access to the electronic publications and integration of the Document and Publication server into international networks and retrieval systems.
- To support scientists of Humboldt University with the creation of publication lines and ejournals.
- Test new forms of publishing.
- Take an active part within the creation of a digital library for the Humboldt University.
- To work in third-party funded projects on practical implementations, testing and evaluation of new and not yet standardised technologies for electronic publishing.

Regular staff as well as project staff and student workers from both institutions: the University Library and the Computer and Media Services of Humboldt University are working together in this group.

## 2 Hard- and Software

Contact persons for the EDOC server are Susanne Dobratz and Bert Wendland.  
For maintenance and technical support Michael Bachmann is responsible.

EDOC as system of hard- and software is part of the technical infrastructure and technology that is used by the Computer and Media Service and the University Library of Humboldt-Universität zu Berlin. The Document and Publication Server (EDOC) has been developed and extended since September 1997. EDOC's technical basis is regularly being updated and adapted to current technical developments and standards.

### **Current Hardware:**

- Server: Sun Workstation (2 x 300 MHz CPUs, 1GB RAM, RAID-System)
- Backup-system: Tivoli Storage Manager (TSM), 2 x IBM H70-7026 (regionally separated) using 2 x IBM 3494 Tape Libraries
- Database-server: Sun Fire V880 (6 x 900 Mhz Ultra SPARC-III, 36 GB)

### **Current Software**

- Operating System: Sun Solaris 2.6
- Sybase Adaptive Server Enterprise
- PHP-frontends (MetaIn, MetaOut, MetaSearch -additional documentations-)
- Apache webserver
- Tomcat
- Cocoon2
- xml, xslt-Processors (Java)

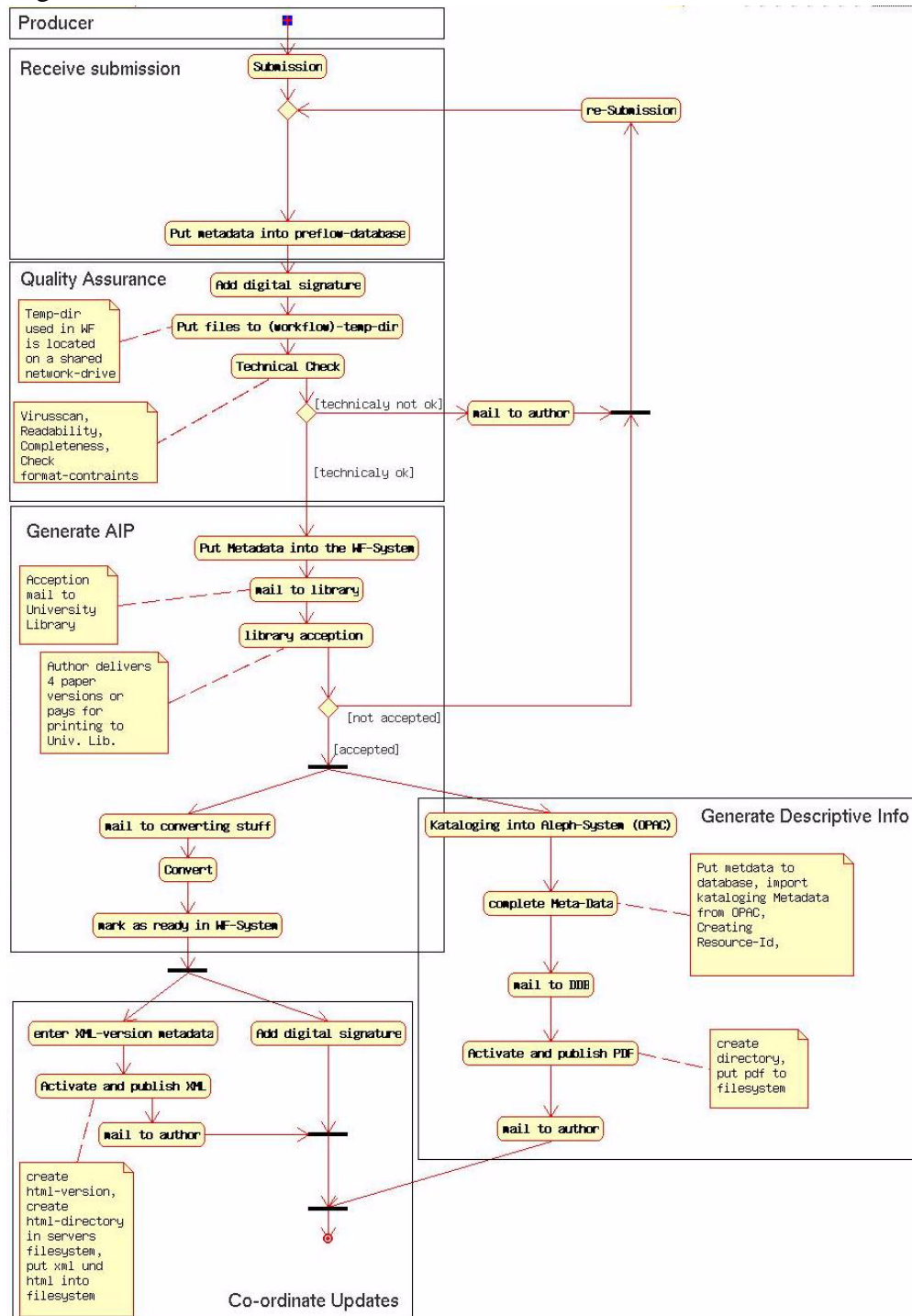
## 3 Edoc mapping against OAIS<sup>2</sup>

---

<sup>2</sup>OAIS, <http://ssdoo.gsfc.nasa.gov/nost/wwwclassic/documents/pdf/CCSDS-650.0-B-1.pdf>, page 4-1

### 3.1 Ingest

Ingest Workflow from edoc-server Humboldt-Universität zu Berlin.



*Graphic 1: Ingest-Workflow*



### **3.1.2 Quality Assurance**

#### **Check (completeness, style)**

- manual check for technical correctness and completeness
- automatic check of template match (check.dot special template, small bugs -> manually revised, big bugs -> document back to author)

### **3.1.3 Generate AIP**

#### **Create XML**

- for theses, journals, reports, proceedings, books
  - converting with XSLT filters for Star- or OpenOffice and Word 2003
- for other digital materials
  - using modified Ebind DTD (XML based) and a text-file (document structure and link list to images) parse to a perlscript
- Plans
  - creating PDF and ebook dynamical with XSL-FO stylesheet

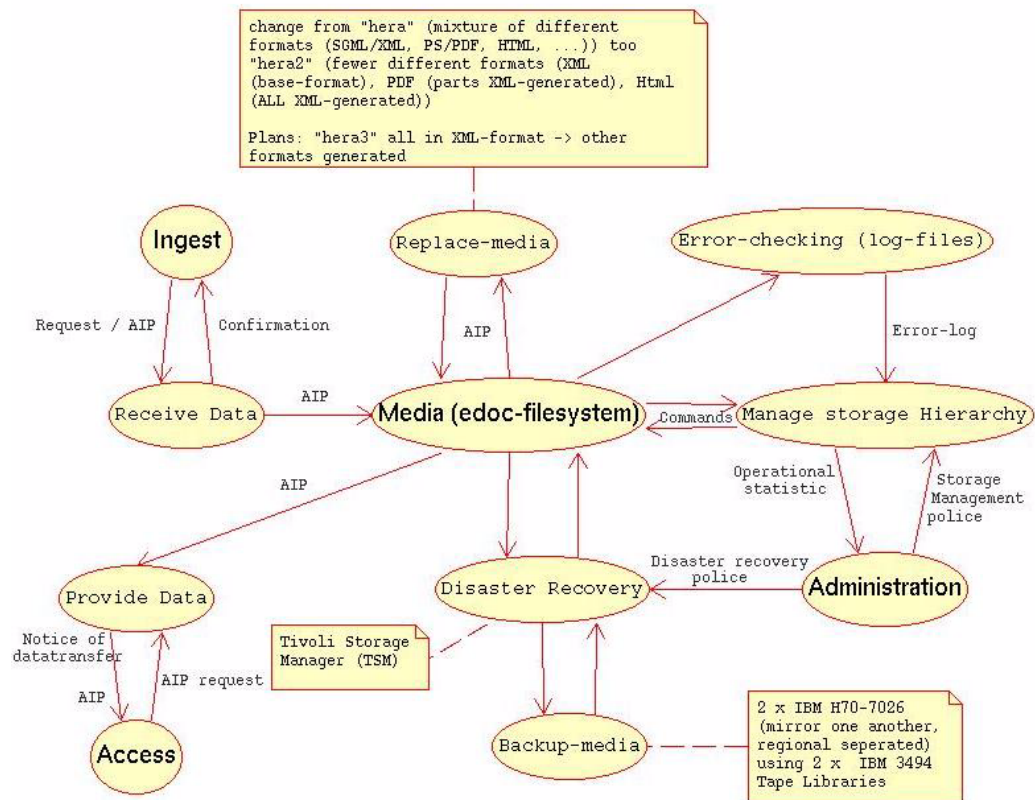
### **3.1.4 Generate Descriptive Information**

- import OPAC entry into edoc-metadata database
- complete the metadata entry in the metadata database
- the metadata entry is accessible via OAI-PMH

### **3.1.5 Coordinate Updates**

- Metadata and content files are inserted into the edoc-system (html version, pdf version, xml version)
- Plans: only XML on the server -> all other formats will be automatically generate

## 3.2 Archival Storage



Graphic 3: archivalmodel based on OAIS

### 3.2.1 Receive Data

- Metadata and content files are manually inserted into the edoc-system
- Plans: automatic transfer based on PHP-/Java- and Shell-script

### 3.2.2 Manage Storage Hierarchy

- files are stored in separate directories
- metadata stored in metadata database

### 3.2.3 Replace Media

- see the above graphic

### 3.2.4 Error Checking

- Server file system check
- Object checked in permanent cycle by MD5 (SSH)

### **3.2.5 Disaster Recovery**

- whole system is managed by the Tivoli Storage Manager (TSM), using the „Backup-media“ (see 3.2.6)

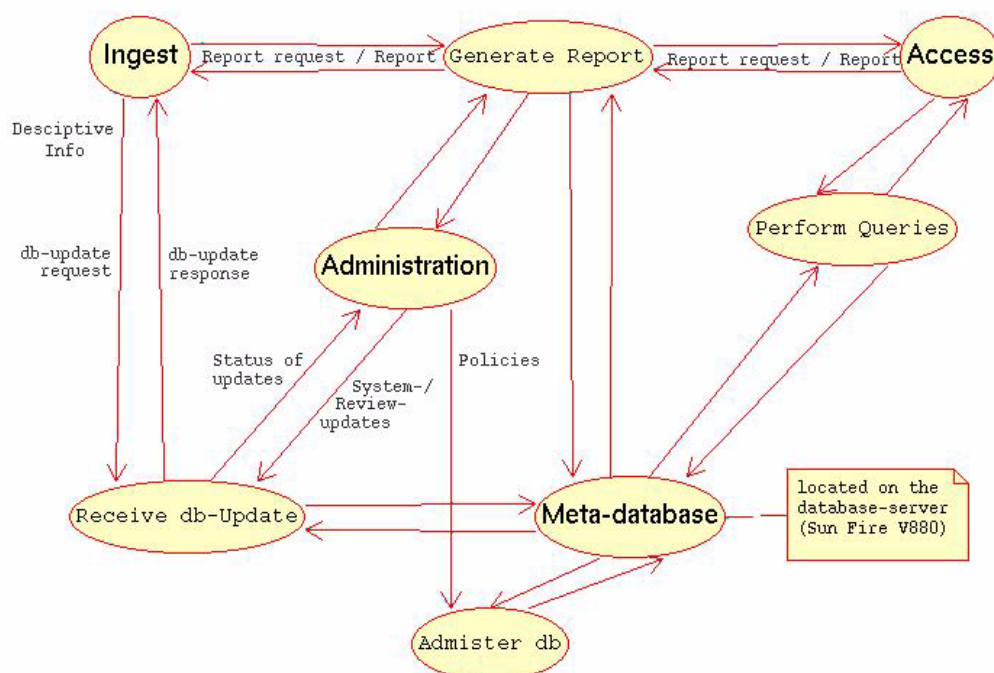
### **3.2.6 Backup-media**

- 2 servers mirror one another, they are regionally separated
- data stored on IBM 3494 Tape Libraries

### **3.2.7 Provide Data**

- Documents accessible via edoc-system

### 3.3 Data Management



Graphic 4: data management based on OAIS

#### 3.3.1 Administer Database

- The input tool „MetaIn“ is responsible for maintaining the integrity of the metadata database.
- The bibliographical records are based on different metadata schemas, e.g. Dublin Core, RAK-WB and ProPrint metadata schema.

#### 3.3.2 Perform Queries

- This is one functionality of “MetaOut“, a request (query) is transferred to the edoc-system and the customer receives the object (file).

#### 3.3.3 Generate Report

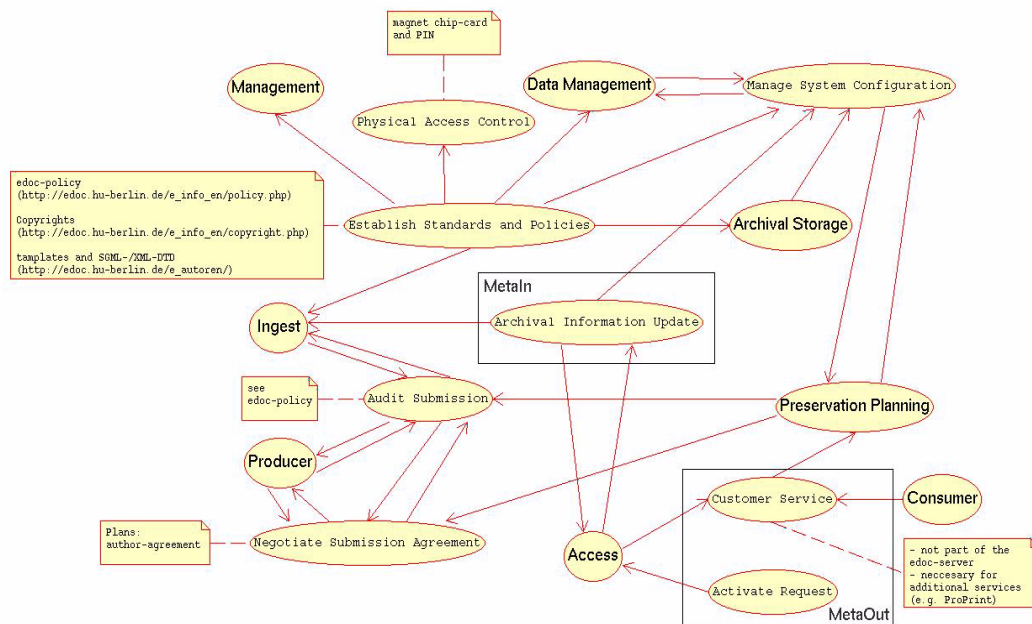
- This is one functionality of „MetaOut“, e.g. fetch number of documents in the database, summaries of archive holdings by categories, alphabetic order, subject order and faculty order.

#### 3.3.4 Receive Database Updates

- This is one functionality of „MetaIn“, function adds, modifies or deletes information in

the Data Management (see 1.1).

### 3.4 Administration



Graphic 5: administration based on OAIS

#### 3.4.1 Negotiate Submission

- solicits desirable archival information for the OAIS and negotiates Submission Agreements with Producers
- is held by e-mail, by telephone and by personal contacts
- Plan
  - author agreement

#### 3.4.2 Manage System Configuration

- function provides system engineering for the archive system to continuously monitor the functionality of the entire archive system and systematically control changes to the configuration
- data not regularly analysed

#### 3.4.3 Archival Information Update

- This is one functionality of „MetaIn“, updating the objects in the archive (see 1.1 and the above graphic).

#### 3.4.4 Physical Access Control



- the physical access is realised via magnet chip-cards and PINs

### **3.4.5 Establish Standards and Policies**

- Electronic Publishing Group
  - develop and establish archive standards and policies
  - develop tools
- CMS intern
  - disaster recovery (TSM, see 3.2.5)
- CMS and Council of Humboldt-University
  - human and budget decision

### **3.4.6 Audit Submission**

- based on the edoc-policy (quality assurance, see 3.1.2)
- the presentation of the content information has not to be changed (the kind of document determines the designated Community)

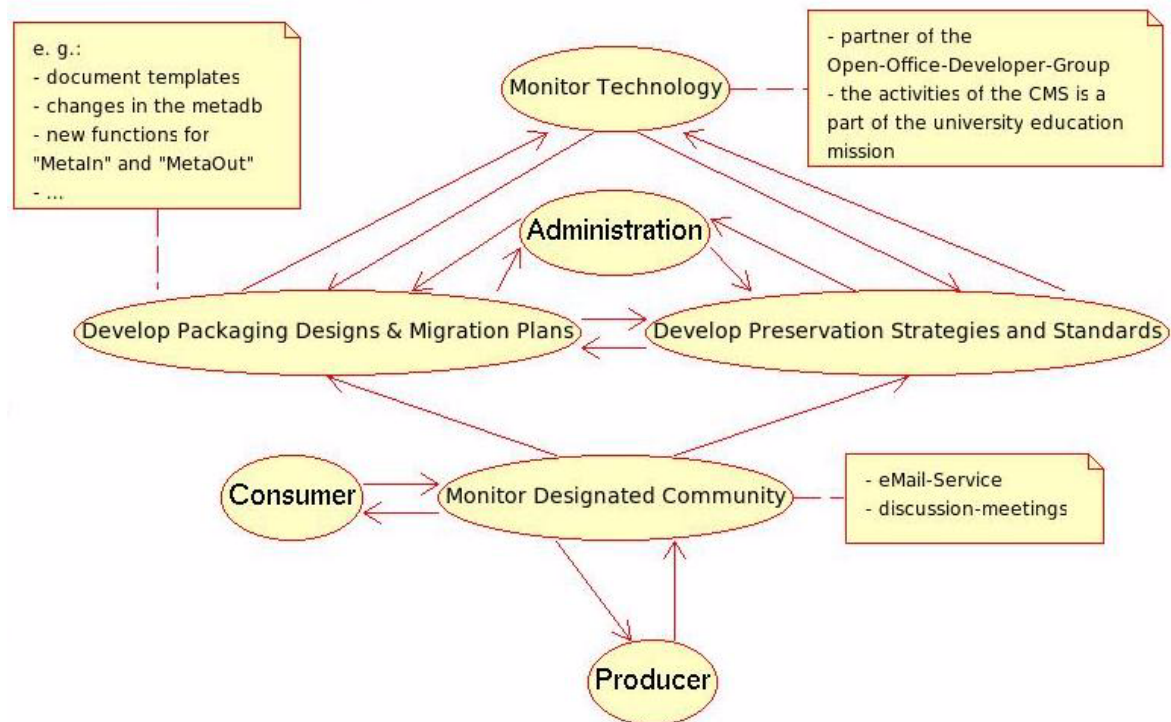
### **3.4.7 Activate Requests**

- this is one functionality of “MetaOut“ (see 1.2 and graphic above)

### **3.4.8 Customer Service**

- this is one functionality of “MetaOut“ (see 1.2 and graphic above)
- it is not part of the edoc-server
- necessary for additional services (e.g. ProPrint)

## 3.5 Preservation Planning



Graphic 6: preservation planning based on OAIS

### 3.5.1 Monitor Designated Community

- e-mail Service
- discussion meetings

### 3.5.2 Monitor Technology

- partner of the OpenOffice-Developer-Group
- the activities of the CMS are part of the university education mission (has to be up-to-date)

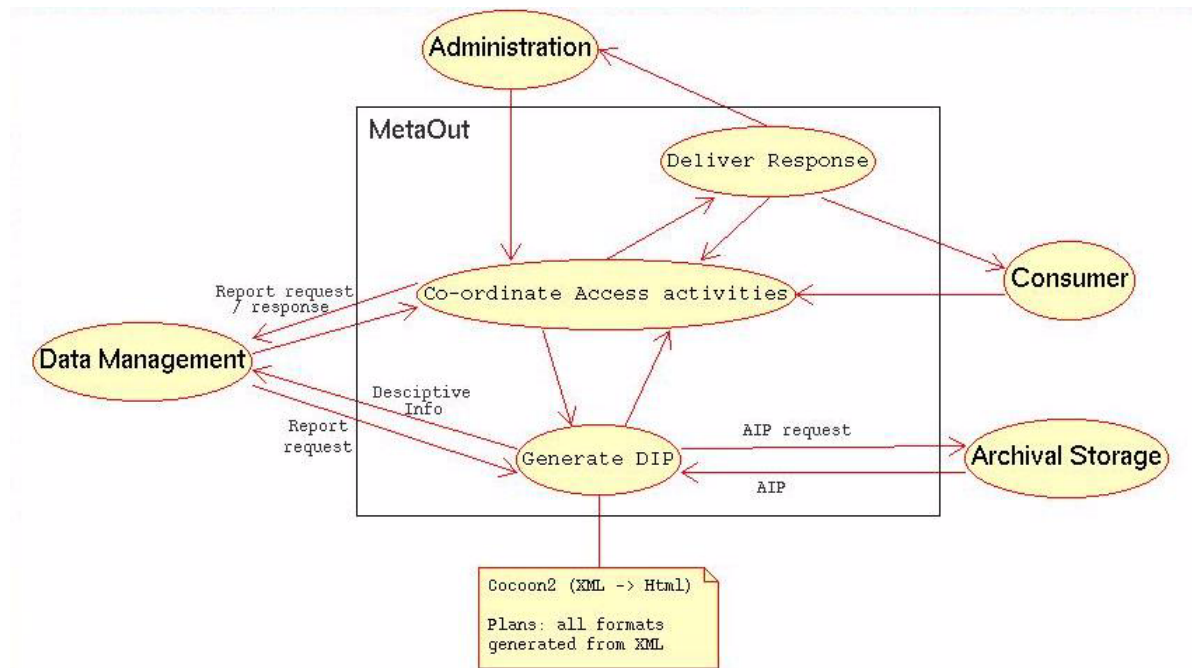
### 3.5.3 Develop Preservation Strategies and Standards

- not a special activity
- only monitoring new technologies (see 3.5.2)

### 3.5.4 Develop Packaging Designs and Migration Plans

- is part of the daily work
- document templates (for the different text-editors, see 3.1.1 „file formats“)
- changes in the metadata database
- new functions for „MetaIn“ and „MetaOut“

## 3.6 Access



Graphic 7: access based on OAIS

All the parts of Access are functionalities of „MetaOut“.

### 3.6.1 Co-ordinate Access Activities

- interface enables sending queries to metadata index and retrieving objects from archive
- records from the edoc-system are linked to the interface using specific object PIDs
- interface works via edoc API methods

### 3.6.2 Generate DIP

- different views/formats of object are generated (e.g. using cocoon2 to transform XML to HTML)
- Plan
  - all formats generated from XML

### 3.6.3 Deliver Response

- requested document is displayed

## 4 Meta-DB and Metadata

## 4.1 Metadata Database

Currently a Relational Database System (SQL, Sybase) is used to store metadata for the repository.

### Features:

- Enabling searching and retrieval of documents stored in the file system
- Generic mapping of DC Scheme to a relational scheme Storing administrative Metadata
- Storing administrative Metadata
- Contains Metadata about itself - dynamic SQL
- core layer (portable)- bibliographical and administrative Metadata, keys, tables
- adaption layer (RDBMS and application specific)- special tables, indices, temp-tables, views, derived attributes, triggers, stored procedures
- changing Metadatastructure is not big deal, also not adapting for other needs - tools easily adaptable
- multilinguality: utf-8 encoding, multilanguage keywords, titles, description

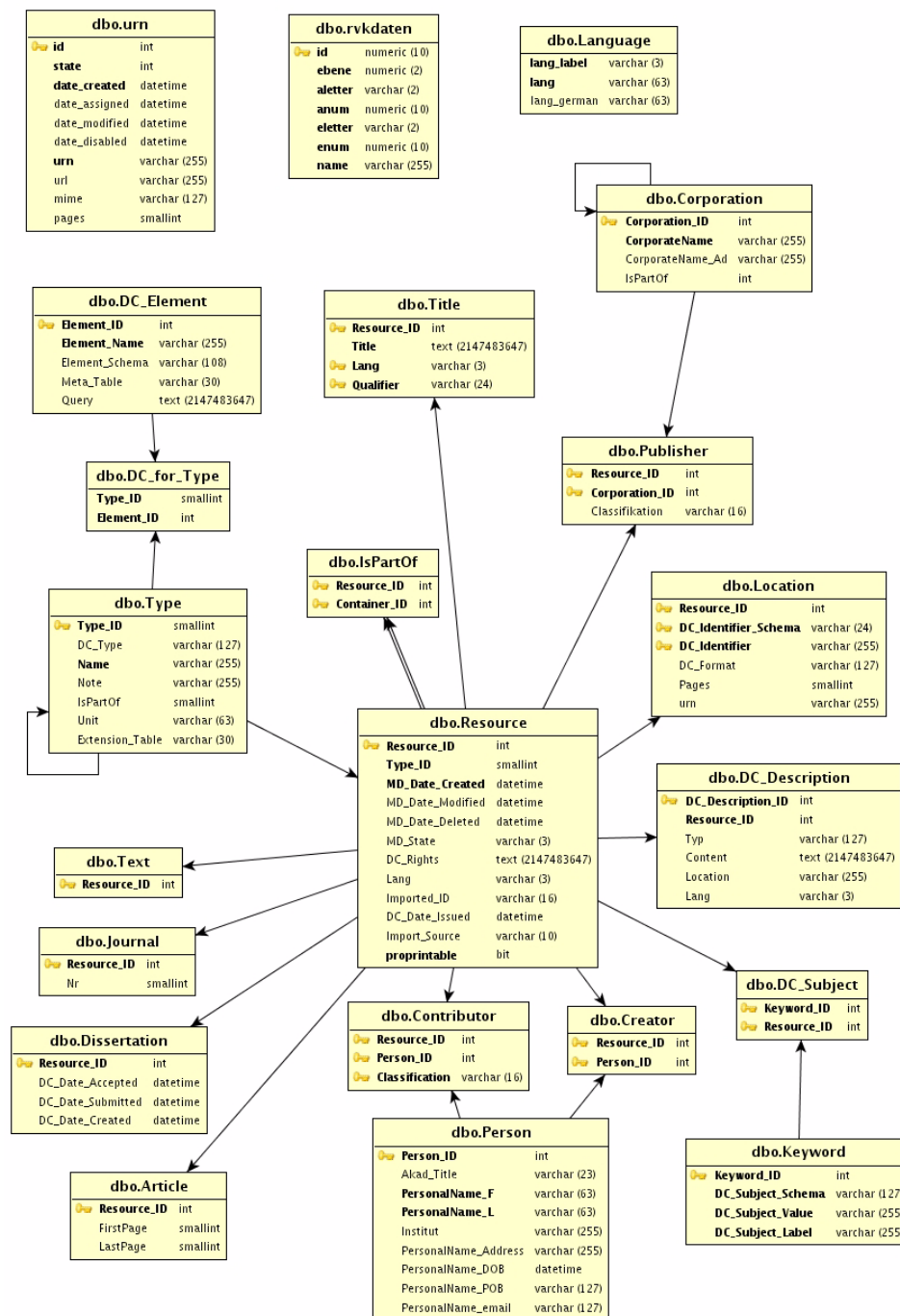
### 4.1.1 Database Structure

The metadata database is an instance of Sybase Adaptive Server Enterprise<sup>5</sup>. The metadata database is used for metadata storage. The metadata format are based on DublinCore, especially METADISS<sup>6</sup> and RAK-WB. The following graphic describes the metadata database schema.

---

<sup>5</sup><http://www.sybase.com/products/informationmanagement/adaptiveserverenterprise>

<sup>6</sup><http://deposit.ddb.de/metadiss.htm>



Graphic 1: Metadb-Schema

The MetaDB has two PHP-frontends the MetaIn and the MetaOut/MetaSearch.

#### 4.1.2 Storing and Finding Data

##### Overall Situation:

- Many sources (SQL, XML, PDF, Word, HTML, Aleph, ... ) each with its own technique requiring different competence
- Complexity results in a lack of understanding what information is available
- Simple questions are translated into complicated SQL-Querys (Joins, Subqueries)
- Exact keyword matching (best for SQL) does not provide all the available information from data sources

- Free form search requires table scans

### **Search Technology:**

- Natural language querying
- Inverted indexing (precise, linguistic, mixed)
- Internet search engines providing context-sensitive searching and concept searching
- Database text searching (custom programming can provide end-user capabilities similar to those available by Internet search engines)
- But! If the target data is highly structured, you may not need search capabilities at all.

### **Issues in Multilinguality**

- Only a first approximation implemented by using utf-8, keywords, titles, descriptions
- Alphabetical ordering still a problem with current Sybase ASE 12.5.1 and utf-8 encoding, next version coming 2005 will hopefully solve the problem
- Multilingual Recognition and Representation
- Cross-Language retrieval (machine translation techniques, knowledge-based techniques (dictionaries, thesauri, ontologies), corpus-based techniques)

## **4.2 Metadata**

### **4.2.1 Dublin Core**

See: <http://dublincore.org/documents/dcmi-terms/>

### **4.2.2 ProPrint Metadata**

See also: <http://edoc.hu-berlin.de/proprint/documentation.xml>

### **The ProPrint Application Profile consists of several namespaces:**

- ProPrint Metadata Schema
- Dublin Core Metadata Element Set, Version 1.1 (dc1.1)
- Dublin Core Qualifiers (dcterms)
- MetaDiss Version 1.4
- DIEPER
- ProPrint Element Set, Version 1.0 (PP1.0)
- ProPrint Qualifiers, Version 1.0 (PPq1.0)

### **4.2.3 MetaDis**

See: <http://deposit.ddb.de/metadiss.htm>

### **4.2.4 OAI-Interface**

If you need metadata please use the OAI-Interface. For more information see <http://edoc.hu-berlin.de/oai/>

## **Chapter 2**

### **Metaln**

#### **Deliverable D1.4 Digital Repository – Germany**

# Inhaltsverzeichnis

1	MetaIn .....	4
1.1	Eigenschaften .....	4
1.2	Installation .....	4
1.3	Die Konfiguration .....	4
1.3.1	Erzeugen des Verzeichnisses .....	4
1.3.2	Die Datei metain.ini .....	4
1.3.3	Überblick über die funktionale Konfiguration .....	5
1.4	XML-Konfiguration der Datenbank Struktur .....	5
1.4.1	Anmerkung zum Sprachgebrauch .....	6
1.4.2	Listen und Datensätze .....	6
1.4.3	Die Datenbank Konfiguration und Eingabemaske .....	6
1.4.4	Eintragen der Listenwerte in die DB .....	6
1.4.5	Beziehungen zwischen den Listen .....	6
1.4.6	Tabellen und Relationen .....	7
1.5	Die XML-Konfiguration der Seiten Struktur (HTML) .....	10
1.5.1	DTD zur Seiten Konfiguration (HTML) .....	11
1.5.1.1	Konfiguration von Formularfeldern .....	14
1.5.2	Der Search Style .....	18
1.5.2.1	DTD des Search Style .....	18
1.5.2.2	Beispiel .....	22
1.5.2.3	Benutzte Klassen .....	23
1.5.3	Der Mail Style .....	23
1.5.3.1	DTD für Style .....	24
1.5.3.2	Beispiel .....	24
1.5.3.3	Benutzte Klassen .....	26
1.5.4	Der Commit Style .....	26
1.5.4.1	DTD für Style .....	27
1.5.4.2	Beispiel .....	28
1.5.4.3	Benutzte Klassen .....	28
1.5.5	Der Multi_in Style .....	29
1.5.5.1	DTD für Style .....	29
1.5.5.2	Beispiel .....	31
1.5.5.3	Benutzte Klassen .....	31
1.5.6	Der Multi_in_to_db Style .....	31
1.5.6.1	DTD für Style .....	32
1.5.6.2	multi_in_to_db Beispiel .....	32
1.5.6.3	Benutzte Klassen .....	33
1.5.7	Der Single_in Style .....	33
1.5.7.1	DTD für Style .....	33
1.5.7.2	Beispiel .....	33
1.5.7.3	Benutzte Klassen .....	34
1.5.8	Der File_import_page Style .....	34
1.5.8.1	DTD für Style .....	34
1.5.8.2	Beispiel .....	34
1.5.8.3	Benutzte Klassen .....	35
1.5.9	Der Plain Style .....	35
1.5.9.1	DTD für Style .....	35
1.5.9.2	Beispiel .....	35
1.5.9.3	Benutzte Klasse .....	35



1.6	Anwendungsdesign .....	35
1.6.1	Der Datenbank-Prozess .....	36
1.6.2	Wichtige Funktionsprinzipien .....	36

# 1 MetaIn

MetaIn ist ein universelles Tool zum Erstellen von Eingabemasken für komplexe Datenbankstrukturen. Das Besondere an MetaIn ist die Konfigurierbarkeit mit Hilfe von XML-Konfigurationsdateien. Damit kann dem Benutzer eine für seine spezielle Aufgabe vereinfachte Sicht auf die Datenbank mit einer speziellen Eingabe präsentiert werden. Der Benutzer wird möglichst linear durch den Eingabeprozess geführt. MetaIn ist in PHP Version 4.3.+ programmiert und lässt sich in einer entsprechenden HTTP-Server Umgebungen betreiben. Als Datenbank ist das DBMS Sybase oder MySQL notwendig.

## 1.1 Eigenschaften

- Der Eingabeprozess ist in der Regel um eine zentrale Tabelle organisiert.
- Der Eingabeprozess ist im Wesentlichen linear, aber frei navigierbar.
- Inserts erfolgen am Ende des Prozesses, wenn alle Daten gesammelt sind, UPDATE und DELETE sofort.
- Eine Formularseite besteht in der Regel aus einem Formular und einer Anzeigeliste der eingegebenen Datensätze.
- XML-Konfigurierbarkeit der Datenbankstruktur und der Seitenstruktur
- Einfach installierbar
- Unterstützung von DBMS Sybase und Mysql, Erweiterung auf weitere möglich
- Automatische Menügenerierung (beliebig viele Ebenen, 30 Seiten pro Ebene)
- Trennung von Ausgabe- und Anwendungslogik
- UTF-8 Unterstützung
- externe Scripte können eingebunden werden
- Weitere Styles samt Konfiguration können leicht eingebunden werden dank geeigneten Frameworks
- Import von Werten aus Dateien
- Datenimport im MARC-Format (für spezielle Tags)
- Template-basierte Mailgenerierung aus den Eingabedaten
- Import von Daten aus der Datenbank über eine Suche

## 1.2 Installation

Kopieren Sie das Programm in ein Web Verzeichnis und passen Sie die Datei `global_settings.inc` an. Folgen Sie den Instruktionen der Beispiel Templates die im Verzeichnis `templates_ger` gespeichert sind.

## 1.3 Die Konfiguration

### 1.3.1 Erzeugen des Verzeichnisses

### 1.3.2 Die Datei `metain.ini`

Diese Datei sollte im selben Verzeichnis gespeichert sein wie `index.php`. Konfigurieren Sie den debugging modus, den Datenbankzugriff, den Pfad zu den Template-Dateien und den Dateinamen der XML-Konfiguration.

### 1.3.3 Überblick über die funktionale Konfiguration

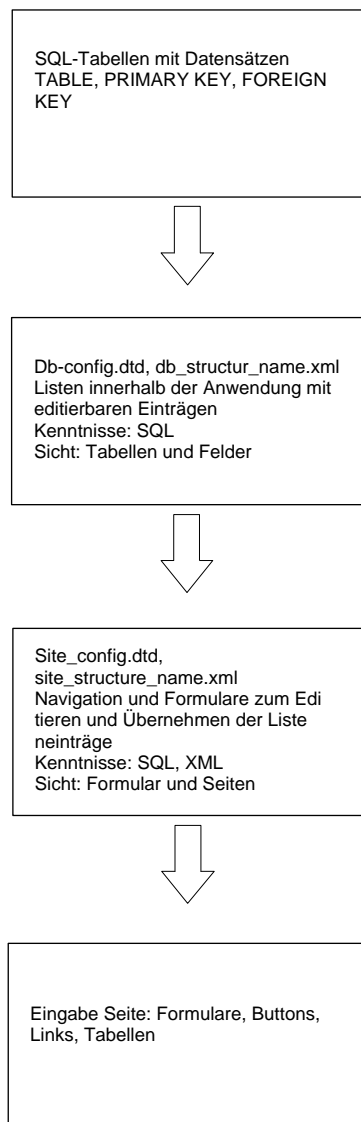
Es besteht ein hoher Grad an Wiederverwendbarkeit bestehender Konfigurationen. Die Möglichkeiten der Gestaltung sind sehr weitreichend, dadurch können auch andere Anwendungen für die Eingabe von nicht bibliographischen Metadaten programmiert werden. Vorteile gegenüber dieser Programmierung: statt der dynamischen Programmierung, es werden nur noch zwei statische Konfigurationsdateien editiert. Programmierkenntnisse entfallen dadurch.

Die Datenbankeingabe besteht aus zwei Ebenen

- Die Struktur der Datenbank
- Die Struktur der Eingabemasken

Zwischen beiden vermittelt die Konfiguration

*Graphic 1: Funktionale Konfiguration - Datenbank zur*



*Benutzereingabe*

### 1.4 XML-Konfiguration der Datenbank Struktur

Die XML-Konfiguration für die Datenbank wird in einer Datei (db\_structure.xml) gespeichert und im Wurzelverzeichnis der Konfiguration abgelegt. Diese Konfiguration ist vollkommen von der Seitenkonfiguration getrennt, die Seitenkonfiguration benutzt aber die in der DB-

Konfiguration verwendeten Bezeichnungen und Nebenbedingungen. Der Name dieser Datei wird in der meta.ini-Datei der Anwendung bekannt gemacht. Die Anwendung parst bei Sessionbeginn diese Datei und erzeugt daraus eine Reihe von Objekten (Listen, Einträge, Links usw.), die als Container für die einzugebenden Daten dienen. Verfällt die Session oder wird Logout vorgenommen verfällt auch die eingelesene Konfiguration. Dies ist insbesondere beim Ändern der Konfiguration zu beachten.

Nach wohldefinierten Aktionen werden die Werte aus den Listen in die DB geschrieben. Dazu muss die Struktur der Datenbankanwendung bekannt sein. Um diese Struktur korrekt zu konfigurieren sind Grundkenntnisse über die Struktur Relationaler Datenbanken (Tabellen, Relationen, Fremdschlüssel, Primärer Schlüssel, Feldtypen) notwendig.

Für eine genaue formale Beschreibung siehe db\_conf.dtd. Bei der Programmierung ist nicht nur auf syntaktische- sondern auch auf die semantische Korrektheit zu achten.

#### **1.4.1 Anmerkung zum Sprachgebrauch**

Tabelle bezieht sich auf die Tabelle in der DB, Liste oder Tabellen-Liste bezieht sich auf die zu einer Tabelle gehörigen Liste von Datensätzen, die gerade bearbeitet werden und sozusagen einen temporären Ausschnitt der jeweiligen Tabelle auf Anwendungsebene darstellen.

#### **1.4.2 Listen und Datensätze**

Diese Listen und ihre Verbindungen bilden ein Abbild eines für die jeweilige Aufgabe relevanten Ausschnitts der Datenbank, in den meisten Fällen wird es nicht nötig sein die gesamte DB-Struktur für MetaIn sichtbar zu machen, aus Performance und Stabilitätsgründen sollte man sich auf den minimalen Ausschnitt beschränken.

Im Wesentlichen werden hier die Tabellen, deren Beziehungen untereinander und die Typen der Felder konfiguriert. Datensätze werden bearbeitet und in diesen Listen gespeichert. Datensätze haben einen Status (INSERT, CHECKED\_INSERT, UPDATE, DELETE), der von der Vorgeschichte des Datensatzes abhängt und entsprechend dem das Schreiben in die Datenbank erfolgt.

#### **1.4.3 Die Datenbank Konfiguration und Eingabemaske**

Teilweise hat die Konfiguration der Datenbank Einfluß auf das GUI (Graphical User Interface). So werden die Typen der Felder aus der Sicht der Eingabemaske betrachtet, dazu gehört die Angabe der für die Darstellung der Input-Felder nötigen Informationen.

#### **1.4.4 Eintragen der Listenwerte in die DB**

Einige Werte der Listeneinträge können erst bei Übernahme in die DB ermittelt werden. So numerische Id's, die als auto\_increment beim Eintragen in die Tabellen automatisch hochgezählt werden sollen. Diese werden dann weitergereicht an die Listen für Relationentabellen. Deshalb ist die Reihenfolge beim Einfügen in die Datenbank zu beachten, siehe auch site\_conf.dtd , final\_insert - active\_tables.

#### **1.4.5 Beziehungen zwischen den Listen**

Tabellen können entweder Daten aus bestimmten Feldern in andere Tabellen exportieren oder aus anderen Tabellen importieren (FOREIGN KEY). Insbesondere gilt dies für Tabellen bzw. Listen, die Relationen repräsentieren.

## 1.4.6 Tabellen und Relationen

Wie kommen Werte aus anderen Tabellen in die Relationen? Bei der Konfiguration der FOREIGN KEYS kann der Modus für den import angegeben werden :

- **choiced** -- der Wert wird über ein drop-down-Menü im Formular vom User selbst gewählt, dabei kann die Werteliste entweder über ein select aus einer anderen Tabelle automatisch generiert werden, per explizite Aufzählung in der Konfiguration angegeben werden oder über eine zugeordnete Suche in eine Tabellen-geschrieben werden, aus der dann die Einträge geholt werden;
- **import** -- hier werden die Daten automatisch aus der zugehörigen Tabelle geholt und die vorhandenen Listeneinträge beschreiben;
- **add** -- hier werden die Daten aus der zugehörigen Tabelle geholt und es werden damit neue Listeneinträge generiert;
- **disabled** -- dieser foreign key soll beim Eintragen ignoriert werden.

Beschreibung der XML-Struktur als DTD:

```
<!-- 25.01.2005 -->
<!ELEMENT db (table | table_link)*>
<!ATTLIST db
name CDATA #IMPLIED>
<!ELEMENT table (field*)>
<!ATTLIST table
name CDATA #REQUIRED
identifier CDATA #IMPLIED
constraint CDATA #IMPLIED>
<!-- constraint ist "no_delete", "new_on_update" (im Augenblick nicht -->
<!-- impl.) oder sowas wie "on_delete|new_on_update" -->
<!-- identifier soll die Möglichkeit geben mehrere kopien einer Tabelle zu
verwalten, für selbstbezügliche Relationen, d.h. später bei den
Felderkonfiguration diese identifier verwenden -->
<!ELEMENT field EMPTY>
<!ATTLIST field
name CDATA #REQUIRED
type (text | int | smallint | insert_date | update_date | delete_date |
datetime) "text"
constraint (null | not_null) "null"
check_on_insert (yes | no) "no"
use_as (show_for_user | id) #IMPLIED>
<!-- alles muss mit date in der datenbank als datetime deklariert sein , --
>
<!-- zumindest in sybase, um where clausel vernünftig zu generieren -->
<!ELEMENT table_link (foreign_key*)>
<!ATTLIST table_link
from_table CDATA #REQUIRED
to_table CDATA #REQUIRED
auto_insert_mode (import | add | disabled) "disabled">
<!-- nur benutzte eintragen Vorsicht bei f-keys von gleicher quelle wie bei
ispartof -->
<!ELEMENT foreign_key EMPTY>
<!ATTLIST foreign_key
to_field CDATA #REQUIRED
from_field CDATA #REQUIRED>
```

Bedeutung der XML-Elemente

Element	Attribut	Bedeutung
db	b2	root-Element der Konfiguration
	name	Name der Datenbank
table		Für Beschreibung der Daten, die für den Zugriff auf eine Datenbanktabelle benötigt werden.
	name	Name der Tabelle wie in der DB
	identifizier	Der Name der Tabelle unter dem die Anwendung die Tabelle intern führt. Dieser Name ist in der Konfiguration der Site zu verwenden. Diesselbe Tabelle kann also mehrmals unter unterschiedlichen identifizier auftauchen, um bspw. mit Selbsbezüglichen Relationen umzugehen (wie IsPartOf o.ä).
field		Angaben um die Eigenschaft der Datenbank-Felder zu beschreiben.
	name	Feldname wie in der DB
	type	<b>Typ des Feldes wie in Bezug auf die Verwendung in MetaIn (default - text):</b> <ul style="list-style-type: none"> <li>• text - für String-Felder (text,varchar,char,...), in SQL mit Quotes</li> <li>• int - für integer-Felder u.ä., in SQL ohne Quotes</li> <li>• datetime - Datumsfeld , in SQL datetime</li> <li>• insert_date - (nur) bei Insert setzt die Anwendung das aktuelle Datum ein, in SQL datetime</li> <li>• update_date - wir bei update aktualisiert, in SQL datetime</li> <li>• delete_date - offen für spezielle Trigger, bspw. wenn ein Datensatz als nicht mehr verwendbar deklariert werden soll, aber in der DB verbleibt, im Augenblick nicht explizit verwendet, bedarf Ausprogrammierung</li> </ul>
	constraint	ob das Feld leer sein darf, zwei Werte "null"-ja und "not_null" - nein, default ist "null"

	use_as	<p><b>Wenn das Feld eine besonders verwendet wird:</b></p> <ul style="list-style-type: none"> <li>• id - wird als eindeutige id (Primary Key) eines Eintrages (Datensatzes) aus der Tabelle benutzt</li> <li>• show_for_user - wenn der Datensatz in Auswahllisten oder ähnlichen Zusammenhängen ´auftauschen soll werden diese Felder dem User explizit angezeigt, um den Datensatz zu identifizieren oder lesbar zu machen.</li> </ul>
	check_on_insert	<p>"yes" oder "no" (default), gibt an, ob vor dem Einfügen (Insert) in der Datenbank ein Feld mit den gleichen Werten in den so gekennzeichneten Feldern schon existiert. Statt einzufügen, wird der schon in der DB vorhandene Datensatz weiter benutzt. Ein Bsp. sind Keywords, die nicht mehrfach in die Datenbank eingetragen werden sollen - statt einen neuen Datensatz zu erzeugen, wird die id des ersten gefundenen Datensatzes mit den übereinstimmenden Werten in als check_on_insert konfigurierten Feldern genommen und bspw. in eine Relationentabelle eingetragen.</p>
table_link		<p>Damit die Werte aus unterschiedlichen Tabellen auf definierte Weise miteinander verbunden werden können wird table_link benutzt.</p>
	from_table	<p>Identifiziert die Tabelle aus der der Wert bezogen wird. Also der Wert aus table.identifiziert.</p>
	to_table	<p>Identifiziert die Tabelle in die der Werte hineingeschrieben wird.</p>

	auto_insert_mode	<p>beschreibt die Übernahme der Werte zu erfolgen hat, denn unterschiedliche Einträge müssen in Relationen zusammengebracht werden, oft geschieht das über extra Relationentabellen:</p> <ul style="list-style-type: none"> <li>• import - der Wert wird in schon vorhandenen Einträge der Liste importiert bevor diese in die entsprechende Tabelle geschrieben werden, Bsp: es sind mehrere URLs in der zur Tabelle Location gehörenden Liste eingegeben worden, bevor man diese aber nach Location schreiben kann muss die Resource_ID aus der Resource-Tabelle übernommen werden auf die sich die URLs beziehen (table_link von Resource nach Location oder foreign key in Location aus Resource), die Angabe auto_insert_mode="import" stellt sicher, dass vor dem Insert der Wert von Resource_ID nachgetragen wird.</li> <li>• add - es werden neue Einträge in der entsprechenden Liste angelegt, Bsp: Es wurden Werte für Herausgeber in die Liste für Corporation eingetragen, Corporation und Resource sind über eine dritte Relationen-Tabelle Publisher verknüpft, es genügt also nicht nur Corporation-Felder einzufügen, danach müssen die Corporation_ID und Resource_ID in Publisher eingesetzt werden, der Link von Corporation nach Publisher muss nun mit auto_insert_mode="add" angegeben werden, damit neue Felder angelegt werden in Publisher, danach werden dann über den Link (auto_insert_mode="import") von Resource(Resource_ID) nach Publisher(Resource_ID) die verbindenden Resource_ID-Werte nachgetragen, dann erfolgt erst der Insert nach Publisher.</li> <li>• disabled - der Link wird ignoriert (manchmal wünschenswert)</li> </ul>
foreign_key		Beziehung zwischen 2 Feldern
	to_field	Zielfeld
	from_field	Quellfeld

## 1.5 Die XML-Konfiguration der Seiten Struktur (HTML)

Diese Beschreibung baut auf der Beschreibung der Konfiguration der Datenbankstruktur für MetaIn und der Überblicksbeschreibung der Konfiguration von MetaIn auf, der Leser sollte mit diesen vertraut sein.



Die Eingabe von zusammenhängenden Daten in eine DB wird als ein gerichteter Vorgang betrachtet bei dem Daten in einzelne, miteinander verbundene Tabellen eingefügt werden. An einer bestimmten Stelle dieses Vorganges erfolgt ein nochmaliges Anzeigen aller in der Anwendung editierten Daten, danach werden die Daten in die DB übertragen. Im Laufe des Eingabeprozesses wird auf schon vorhandene Hilfsdaten zurückgegriffen, die über eine Suche ermittelt werden oder über extra Seiten eingegeben werden können. Weiterhin wird es eine Suche geben, um schon eingebene Daten in die Anwendung zu holen und editieren zu können. Die Anforderungen an die Eingabe, die sich aus unterschiedlichen Arten von Tabellen und Tabellen-Beziehungen ergeben, können mit der Auswahl von unterschiedlichen, fest definierten Stilen für die Gestaltung der Eingabemaske berücksichtigt werden. Obwohl es theoretisch möglich ist mehrere Eingabeprozesse innerhalb einer site zu konfigurieren, sollte darauf verzichtet werden, da die Einträge in den Tabellenlisten nicht mehr eindeutig zugeordnet werden können und sich unerwünschte Überschneidungen ergeben könnten. Die Konfiguration erfolgt Seitenweise (page), es gilt - eine Seite ein Stil. Zu jeder Seite gehört eine eindeutige id, die aus einem Wort aus kleinen Buchstaben (und Ziffern) besteht. Es gibt zwei unabhängig konfigurierbare Navigationsmöglichkeiten - das Menü (global) und Links (lokal), über die nächste, vorgehende und zugeordnete Seiten erreicht werden können. Die Lage im Menü wird automatisch aus dem id-Wort ermittelt. Die Anzahl der Buchstaben im Wort gibt die Tiefe der Lage im Menü-Baum an. Innerhalb einer Baumebene (gleicher Ast, gleiche Tiefe) wird die Reihenfolge alphabetisch ermittelt.

```
a-
| -aa
| -ab-
| -aba
| -abb
```

```
b-
| -ba
| -bx
```

```
d-
| -da-
| -dac-
| -dacb
```

...

Styles : Im Augenblick werden mehrere verschiedene Styles mit unterschiedlicher Funktionalität unterstützt, weitere können von PHP-Programmierern mit Kenntnissen in OOP hinzugefügt werden. Die unterschiedlichen Anforderungen an die Styles erfordern jedoch unterschiedliche Konfigurationselemente, so dass auch der XML-Konfiguration-Parser und die dtd entsprechend erweitert werden müssten.

### 1.5.1 DTD zur Seiten Konfiguration (HTML)

Die entsprechende DTD ist hier

Allgemeine Elemente (für Einzelheiten bitte die Styles beachten):

Allgemeine XML-Elemente für die Seitenkonfiguration

Element	Attribut	Bedeutung
site		root-Element der Konfiguration

	name	Name der Site
	start_id	Page-id to start with by default
	clear_tables_from_commit_page	Page-id of commit-style page, to clear when new-Button is clicked or NEW-Action appears. The tables-lists mentioned in that commit-page configuration will be cleared.
	global_dataset_identifier_table	Name der Tabelle, deren Id benutzt wird, um den Eingabestatus anzuzeigen. Das Id-Feld sollte in der DB-Konfig. gekennzeichnet sein. Die Eingabe erfolgt meistens um eine zentrale Tabelle (global_dataset_label_table).
	global_dataset_label_table	Angabe der Tabelle aus der Felder verwendet werden, um den aktuellen Hauptdatensatz für Menschen erkennbar zu machen.
	global_dataset_label_field	Das Feld zu global_dataset_label_table.
	global_dataset_label_field_label	Wie soll man die Ausgabe des Feldwertes für den User benennen.
page		Container für Seiten-Konfiguration
	id	Die id gemäß den oben beschriebenen Regeln, um die Baumartige Navigation zu ermöglichen.
	name	Name der Seite wird in Navigation angezeigt
	label	Überschrift die auf der Seite oben erscheint
	prev	lokale Navigation, vorherige Seite
	next	lokale Navigation, nächste Seite
	dual	verbundene Seite, z.B. eine Suche, die mit einem Formular in Verbindung steht, ein Abzweig
	help_file	Pfad zu einem HTML-Text-File, das als Hilfe eingeblendet wird

value		allgemeiner Wert, hat zwei Repräsentierungen eine für den User und eine für die interne Verarbeitung
	value	der Wert wie er intern verarbeitet wird und wie er in die DB kommt oder aus der DB kommt, bspw. Keyword_ID
	label	Anzeigewerte von Value wie der User ihn zu sehen bekommt, bspw. das explizit ausgeschriebene Keyword
select		eine Select Anfrage an die DB, wie sie z.B. erfolgt, um vorgegebene Werte für eine Auswahlliste aus der DB zu holen
	id_field	Name des Feldes aus der Resultatmenge, das als eindeutiger Identifier für den Wert steht ähnlich wie bei value.value, der Rest der Werte der restlichen Felder wird zusammengefügt und dem Benutzer zur Leserlichkeit angezeigt. Ein Bsp.: '<select id_field="myid">SELECT Resource_ID as myid, Title as zeige_mich_an from Title</select>' - intern wird die Resource_ID verarbeitet, der User wählt oder sieht nach dem angezeigten Titel.
table		Angaben über eine Tabelle (Liste)
	name	Identifier der Tabellenliste wie in der DB-Konf. verwendet im Kontext von import_tables oder Name der Tabelle in der DB falls in Suchquery verwendet.
	alias	Kontext von Suchabfrage: Alias wie in SQL-Abfragen verwendet: "from Resource R" - R ist der Alias
	level	Kontext von import_tables: Ordnung der Tabelle in der Auslesehierarchie (Reihenfolge) aus der Datenbank, bspw. muss eine unabhängige Tabell in Level niedriger sein, da vorher ausgelesen, als eine abhängige die einen Fremdschlüssel benutzt
	action	<p>Aktion die beim Auslesen und Füllen der Daten in die Listen erfolgen soll:</p> <ul style="list-style-type: none"> <li>• CLEAR_IMPORT - leere die Liste und importiere die neuen Datensätze aus der Datenbank (default)</li> <li>• IMPORT - importiere nur, belasse vorhandenes in der Liste</li> </ul>

		<ul style="list-style-type: none"> <li>• CLEAR - leere nur die Liste</li> </ul>
--	--	---------------------------------------------------------------------------------

### 1.5.1.1 Konfiguration von Formularfeldern

Es wird die Konfiguration von Eingabefeldern für Formulare beschrieben. Diese werden in allen Styles verwendet, wo Daten eingegeben werden (single\_in, multi\_in, multi\_in\_to\_db).

#### XML-Konfiguration von Formularfeldern

Element	Attribut	Bedeutung
date		Feld für die Datumseingabe
	table	Tabellenidentifizier für den Eingabewert
	field	Name des Feldes für den Eingabewert
	label	Bezeichner des Feldes für die Anzeige
	hidden	verstecken? "yes" oder "no"(default), dient der versteckten Übergabe von Werten, bspw. vordefinierte Werte, die vorher fest stehen
	required	Eigabe wird verlangt - "yes" oder "no" (default)
	readonly	wird angezeigt, aber editieren nicht möglich - "yes" oder "no" (default)
text		Textfeld Mit child-Element value kann ein vordefinierter Wert angegeben werden, Verwendung im Zusammenhang mit hidden="yes"
	Attribute wie bei date	
textarea		Textfeld (bei Bedarf mehrzeilig) Mit child-Element value kann ein vordefinierter Wert angegeben werden.

	Attribute table, field, label, readonly, required wie bei date	
	parseable	yes: Der Eingabetext wird zeilenweise zerlegt, um gleichzeitig mehrere Werte zu liefern. no: default
option		Anzeige eines Drop- Down- Menüs Die gewünschten Einträge werden explizit in der Konfiguration angegeben. Siehe auch Element value in Tabelle "Allgemeine XML- Elemente für die Seitenkonfiguration".
	Attribute table, field, label wie bei date	
	multiple	yes: mehrere Items gleichzeitig auswählbar no: default
option_by_ select		Auswahlliste Siehe Element select in Tabelle "Allgemeine XML- Elemente für die Seitenkonfiguration".
	Attribute wie bei option	
parameter		Liste von Parametern Diese Liste dient dazu unterschiedliche Eingabemöglichkeiten für eine Tabelle in einem Formular zusammenzufassen (siehe Beispiel).
	table, field, label wie bei date	
dependend		Ein vom Parameterfeld abhängiges Feld.
	table, field, label wie bei date	
dependend _value		Eine einem Parameterwert zugeordnete Art des Eingabefeldes
	parameter_value	Zuordnung der Anzeige zu einem Parameterwert

	style	Anzeigeart des Feldes - text, textarea, option, option_by_select, file_import. Diese werden analog zu den explizit definierbaren Feldern verwendet, aber im Attribut angegeben.
	parseable, multiple analog zu textarea und option und in abhängig vom style-Attribut zu verwenden.	
file_import		Ermöglicht das Hochladen einer Datei verbunden mit der automatischen Übernahme des Inhaltes in ein Eingabefeld
	table, field, label, required, readonly, parseable analog zu oberen Beschreibungen	
option_by_import		Tabelle, aus der importiert werden soll Dieses Element wurde bisher noch nicht benötigt.
	Attribute wie bei option	
import_by_search		Möglichkeit, aus dem Formularfeld direkt eine Suche für den Eintrag zu starten, Weiterentwicklung von option_by_import Dieses Element wurde bisher noch nicht benötigt.

Hinweis zur Programmierung: Die Felderobjekte werden aus der Konfiguration gelesen und in */metain-bin/gui/FormFieldStyleFactory.class.php* erzeugt.

Graphic 2: Ein parametrisiertes  
Formular

Beispiel für die Benutzung von parameterabhängigen Formularfeldern:

```
<page id="h" name="Keywords" label="Schlüsselwörter" prev="g" next="i"
help_file="../../help/keywords.hlp">
<multi_in constraint="no_edit">
<parameter table="Keyword" field="DC_Subject_Schema"
label="Klassifikation">
<value value="ger" label="deutsch"/>
<value value="eng" label="englisch"/>
<value value="DDC" label="DDC"/>
<value value="DNB" label="DNB"/>
<value value="RVK" label="RVK"/>
</parameter>
<dependend table="Keyword" field="DC_Subject_Value"
label="Schlüsselwörter">
<dependend_value parameter_value="ger" label="deutsch" style="file_import"
parseable="yes"/>
<dependend_value parameter_value="eng" label="englisch" style="file_import"
parseable="yes"/>
<dependend_value parameter_value="DNB" label="DNB" style="option_by_select"
multiple="yes">
<select id_field="id">SELECT
DC_Subject_Value as id, DC_Subject_Value, DC_Subject_Label as label from
Keyword
where DC_Subject_Schema="DNB" order by DC_Subject_Value</select>
</dependend_value>
<dependend_value parameter_value="DDC" label="DDC" style="option_by_select"
multiple="yes">
<select id_field="id">SELECT
```

```

DC_Subject_Value as id, DC_Subject_Value, DC_Subject_Label as label from
Keyword
where DC_Subject_Schema="DDC" order by DC_Subject_Value</select>
</dependend_value>
<dependend_value parameter_value="RVK" label="RVK" style="file_import"
parseable="yes"/>
</dependend>
</multi_in>
</page>

```

## 1.5.2 Der Search Style

Dieser Style ermöglicht die Konfiguration einer Suche und enthält ebenfalls eine Import-Funktionalität für Datensätze aus der DB in die konfigurierten Tabellenlisten, um die Werte bearbeiten zu können.

Die Konfiguration erfordert im Wesentlichen drei Angaben: SQL-Query an die Datenbank, Suchfelder für Anzeige und Abfrage in der DB und Listen, in die Daten nach Auswahl eines gefundenen Datensatzes importiert werden sollen, um diesen zu editieren.

Graphic 3: Der Search

The screenshot shows a web application titled "edoc - Metadaten: Test". It has a navigation bar with "Logout", "Neu", and "Hilfe". A left sidebar contains a menu with "File", "Resource", "Titel", "Autor", "Beteiligte", "URL", "Abstract", "Keywords", "Commit", and "Suche und Import" (which is highlighted). The main content area shows the action "Aktion: Datensatz editieren mit Id:" and the title "Titel: titel". Below this is a section titled "Nach Testeinträgen suchen" with search fields for "Titel:" and "DDC:". There are buttons for "SUCHEN" and "FORMULAR LÖSCHEN". A link "Zurück zu ALEPH-Datei importieren" is also present. A table indicates "4 Einträge wurden gefunden" with the following entries:

Titel	
20443, test um ddc einträge zu generieren	ÜBERNEHMEN
20627, Gangentwicklung und Bewegungswahrnehmung im Hüftgelenk in der Rehabilitation nach TEP-Implantation bei Dysplasiekoxarthrose	ÜBERNEHMEN
20632, testtitel	ÜBERNEHMEN
20666, Sprach-Test	ÜBERNEHMEN

Style

### 1.5.2.1 DTD des Search Style

```

<!ELEMENT search (import_tables,result_query,search_field*) >
<!ATTLIST search
result_style (import_single|import_muliple) "import_single"
max_output_per_page CDATA #IMPLIED >

<!ELEMENT result_query (field*,table*,condition*,tail?)>
<!ATTLIST result_query
id_field CDATA #REQUIRED
id_table_identifier CDATA #REQUIRED

```



```

link_field CDATA #IMPLIED >

<!ELEMENT import_tables (table+)>

<!ELEMENT condition EMPTY>
<!ATTLIST condition
relation CDATA #REQUIRED >

<!ELEMENT field EMPTY >
<!ATTLIST field
alias CDATA #IMPLIED
name CDATA #REQUIRED>

<!ELEMENT tail EMPTY >
<!ATTLIST tail
clause CDATA #REQUIRED>

<!ELEMENT search_field
(condition*,(search_text|search_option|search_option_by_select)) >
<!ATTLIST search_field
search_mode (match_part|match_part_beginning|match_whole) "match_part"
field CDATA #REQUIRED>

<!ELEMENT search_text (value?)>
<!ATTLIST search_text
label CDATA #IMPLIED
hidden (yes|no) "no"
required (yes|no) "no"
parseable (yes|no) "no">

<!ELEMENT search_option (value*)>
<!ATTLIST search_option
label CDATA #IMPLIED
multiple (yes|no) "no">

<!ELEMENT search_option_by_select (select)>
<!ATTLIST search_option_by_select
label CDATA #IMPLIED
multiple (yes|no) "no">

<!ELEMENT table EMPTY >
<!ATTLIST table
name CDATA #REQUIRED
alias CDATA #IMPLIED
level CDATA #IMPLIED
action (IMPORT|CLEAR|CLEAR_IMPORT) "CLEAR_IMPORT">
>

```

### Search Style - XML-Elemente

Element	Attribut	Bedeutung
search		Container für Search Konfiguration

	result_style	<b>Art der Ausgabe der Suchergebnisse</b> <ul style="list-style-type: none"> <li>import_single: nur ein gefundener Datensatz kann auf importiert werden (default)</li> <li>import_multiple: genau ein Datensatz kann importiert werden</li> </ul>
	max_output_per_page	Anzahl der gefundenen Datensätze, die ausgegeben werden sollen (paging).
import_tables		Container für table-Elemente Tabellen, die in der DB-Konfiguration konfiguriert wurden und in deren zugehörige Listen der gewählte Datensatz und alle abhängigen DSe für die Bearbeitung über die Formulare importiert werden sollen.
table		Angaben über eine Tabelle
	name	Identifizier der Tabellenliste wie in der DB-Konfiguration Wird verwendet im Kontext von import_tables.
	alias	Kontext von Suchabfrage Alias wie in SQL-Abfragen verwendet: "from Resource R" - R ist der Alias
	level	Kontext von import_tables Ordnung der Tabelle in der Auslesehierarchie (Reihenfolge) aus der Datenbank, z.B. muss eine unabhängige Tabell in Level niedriger sein als eine abhängige, die einen Fremdschlüssel benutzt.
	action	<b>Aktion, die beim Auslesen und Füllen der Daten in die Listen erfolgen soll</b> <ul style="list-style-type: none"> <li>CLEAR_IMPORT: Leere die Liste und importiere die neuen Datensätze aus der Datenbank (default)!</li> <li>IMPORT: Importiere nur, belasse vorhandenes in der Liste!</li> <li>CLEAR: Leere nur die Liste!</li> </ul>
result_query		Konfiguration der SQL-Abfrage für die Suche, liefert das Resultat
	id_field	Bezeichnung (alias) des als ID zu benutzenden Feldes aus der Resultatmenge als Antwort auf die SQL-Abfrage, z.B. R.Resource as id , dann id_field="id"

	id_table_identifizier	Identifizier der Liste für die Tabelle zu der Id, wie in der DB-Konfiguration angegeben
	link_field	Falls ein Link in der Resultatmenge generiert werden soll (z.B. auf die Frontpage), dann analog zu id_field den Bezeichner für das Feld mit dem Link aus der Resultatmenge der Abfrage wählen.
field		Angaben darüber, wie ein Feld einer DB- Tabelle in der Query benutzt werden soll
	alias	alias für das Feld, z.B. "Titel as Titel"
	name	Name des Feldes wie in der Datenbank, auch mit Tabellen-Alias, z.B. "select R.Resource_ID as id from Resource R ..." - name="R.Resource_ID"
condition		Wird verwendet, um join und andere where-Bedingungen anzugeben. Diese werden im Kontext von search_field nur benutzt, wenn tatsächlich gebraucht.
	relation	String wird in die where-Klausel eingebaut
tail		Alles, was an die Query angehängt wird wie "order by" u.ä.
	clause	Anzuhängende String
search_field		Suchfeld Ausgabe als html und Art der Verwendung in der Suchabfrage
	search_mode	<p>Modus, in welchem die Eingabe in der DB-Feldern gesucht wird</p> <ul style="list-style-type: none"> <li>• match_part: innerhalb eines String-oder Textfelds, entspricht "like '%suchwort%'" (default)</li> <li>• match_part_beginnig: als Anfangszeichenkette eines String-oder Textfelds, entspricht "like 'suchwort%'"</li> <li>• match_whole: als Ganzes eines String-oder Textfelds, genaue Übereinstimmung, entspricht "like 'suchwort'"</li> <li>• match_integer: als Zahl, entspricht "=suchzahl"</li> </ul>
	field	Name des Datenbankfelds in Übereinstimmung mit der Gesamtquery (z.B. field="T.Title")

search_text		HTML-Sucheingabe als Textfeld
	label	Anzuzeigender Eingabefeld-Name
	parseable	yes: Prüfung auf vorhandene Leerzeichen no: default
	required	Angabe ist obligatorisch
search_option		Erzeugung einer Drop-Down-Liste
	multiple	yes: Mehrfachauswahl no: default
	label	Anzuzeigender Name
search_option _by_select		Erzeugung einer Drop-Down-Liste Die Einträge werden über eine select- Anfrage ermittelt (siehe select-Beschreibung oben).
	label	Siehe search_option
	multiple	Siehe search_option

### 1.5.2.2 Beispiel

```

<page id="k" name="Suche und Import" label="Nach Testeinträgen suchen"
dual="b" help_file="../help/search_import.hlp">
<search_result_style="import_single">
<import_tables>
<table name="Resource" level="0" action="CLEAR_IMPORT"/>
<!-- loesche alte Inhalte und importiere Daten aus db, ist default auch
weglassen-->
<table name="Creator" level="1"/>
<table name="Contributor" level="1"/>
<table name="Dissertation" level="1"/>
<table name="Title" level="1"/>
<table name="DC_Description" level="1"/>
<table name="Location" level="1"/>
<table name="DC_Subject" level="1"/>
<table name="Keyword" level="2"/>
<table name="Person" level="2"/>
<table name="C" level="2"/>
</import_tables>
<result_query id_field="Resource_ID" id_table_identifier="Resource">
<field name="R.Resource_ID" alias="Resource_ID"/>
<field name="CONVERT(varchar,R.Resource_ID)+'',
'+CONVERT(char(255),T.Title)+' ' " alias="Titel"/>
<table name="Resource" alias="R"/>

```

```

<table name="Title" alias="T"/>
<!-- man schreibe alle Tabellen hier hinein -->
<!-- condition relation="R.Type_ID=10" / -->
<!-- dissertationen type_id -->
<condition relation="T.Resource_ID=R.Resource_ID and R.Type_ID=122"/>
<tail clause="order by Titel"/>
</result_query>
<search_field field="R.Resource_ID" search_mode="match_integer">
<search_text label="ID" />
</search_field>
<search_field field="T.Title">
<search_text label="Titel" parseable="yes"/>
</search_field>
<search_field field="K.DC_Subject_Value" search_mode="match_whole">
<table name="DC_Subject" alias="S"/>
<table name="Keyword" alias="K"/>
<condition relation="S.Resource_ID=R.Resource_ID and
S.Keyword_ID=K.Keyword_ID and K.DC_Subject_Schema='DDC'"/>
<search_option_by_select label="DDC">
<select id_field="id">SELECT DC_Subject_Value as id, DC_Subject_Label
as label from Keyword where DC_Subject_Schema='DDC'</select>
</search_option_by_select>
</search_field>
</search>
</page>

```


### 1.5.2.3 Benutzte Klassen

- /metain-bin/search\_modul/\*
- /metain-bin/styles/SearchData.class.php
- /metain-bin/styles/SearchPageStyle.class.php
- /metain-bin/styles/SearchItem.class.php

### 1.5.3 Der Mail Style

Dieser Style gibt die Möglichkeit, eine E-mail mit Daten aus dem editierten Datensatz zu versenden. Absender, Empfänger, Betreff und der Inhalt sind in der Konfiguration anzugeben. Die E-mail kann nur verschickt werden, wenn der Datensatz in der Datenbank vorkommt. Für den E-mail-Text und die Betreff-Zeile wird eine einfache Template-Syntax benutzt, um Werte aus der Datenbank übernehmen zu können. Die Erstellung dieser Templates verlangt das Verständnis von SQL und der Tabelle metadb.DC\_Element.

*Graphic 4:* Der Mail  
Style


**edoc - Metadaten: Test**

Logout
Neu
Hilfe

File
Resource
Titel
Autor
Beteiligte
URL
Abstract
Keywords
Commit
Mail
Suche und Import

**Aktion:** Datensatz editieren mit Id: 20443  
**Titel:** test um ddc einträge zu generieren  
  
**Mail an DDB**

Empfänger: konstantin@rekk.de  
Betreff: L6000-0473/H5//000  
Inhalt:

```

<META NAME="DC.Publisher.CorporateName" CONTENT="Humboldt-Universität zu Berlin,
Universität&uml;tsbibliothek">
<META NAME="DC.Publisher.CorporateName.Address" CONTENT="D-10099 Berlin">
<META NAME="DDB.Contact.ID" CONTENT="L6000-0473">
<META NAME="DDB.Contact" CONTENT="dagmar=beyer@ub.hu-berlin.de">
<META NAME="DC.Type" CONTENT="IIIIIIIIITESTIIIIIIII">
<META NAME="DC.Creator.PersonalName" CONTENT="">
<META NAME="DC.Title" LANG="ger" CONTENT="test um ddc einträge zu generieren">
<META NAME="DC.Language" SCHEME="ISO639-2" CONTENT="ger">
<META NAME="DC.Format" SCHEME="IMT" CONTENT="application/pdf">
<META NAME="DC.Contributor.CorporateName" CONTENT="">
<META NAME="DC.Date.Accepted" SCHEME="W3CDTF" CONTENT="--">
<META NAME="DC.Subject" SCHEME="DDC-Sachgruppe" CONTENT="000">
<META NAME="DC.Subject" SCHEME="Freitext" LANG="ger" CONTENT="">
<META NAME="DC.Subject" SCHEME="Freitext" LANG="eng" CONTENT="">
<META NAME="DC.Description" SCHEME="Freitext" LANG="ger" CONTENT="">
<META NAME="DC.Description" SCHEME="Freitext" LANG="eng" CONTENT="">
<META NAME="DC.Identifier" SCHEME="URL" CONTENT="">
<META NAME="DC.Identifier" SCHEME="URN" CONTENT="">
<META NAME="DC.Server" CONTENT="Humboldt-Universität zu Berlin, Universität&uml;tsbibliothek">

```

### 1.5.3.1 DTD für Style

```

<!ELEMENT mail (mail_data?)>
<!ELEMENT mail_data EMPTY>
<!ATTLIST
to CDATA #REQUIRED
from CDATA #REQUIRED
subject_template CDATA #REQUIRED
content_template_file CDATA #REQUIRED>

```

### Mail Style - XML-Elemente

Element	Attribut	Bedeutung
mail_data		root-Element der Konfiguration
	to	Empfänger
	from	Absender
	subject_template	String für Betreff
	content_template_file	Pfad zum Template-File für den Text der E-mail

### 1.5.3.2 Beispiel

```

<page id="j" name="Mail" label="Mail an DDB" >
<mail>
<mail_data to="info@ddb.de"
from="edoc@cms.hu-berlin.de"

```

```

subject_template="L6000-0473/HS/%name=DC.Creator.PersonalName%/%
name=DC.Subject&scheme=DDC%"
content_template_file="ddb_email_test.tpl" />
</mail>
</page>

```

## Template Beispiel

```

<META NAME="DC.Publisher.CorporateName"
CONTENT="Humboldt-Universit&auml;t zu Berlin,
Universit&auml;tsbibliothek">
<META NAME="DC.Publisher.CorporateName.Address" CONTENT="D-10099 Berlin">
<META NAME="DDB.Contact.ID" CONTENT="L6000-0473">
<META NAME="DDB.Contact" CONTENT="dagmar=beyer@ub.hu-berlin.de">
<META NAME="DC.Type" CONTENT="!!!!!!!!!!!!TEST!!!!!!!!!!!!">
<META NAME="DC.Creator.PersonalName"
CONTENT="%name=DC.Creator.PersonalName%">
<META NAME="DC.Title" LANG="%name=DC.Title&query_field=Lang%"
CONTENT="%name=DC.Title%">
<META NAME="DC.Language" SCHEME="ISO639-2" CONTENT="%name=DC.Language%">
<META NAME="DC.Format" SCHEME="IMT" CONTENT="application/pdf">
<META NAME="DC.Contributor.CorporateName"
CONTENT="%name=DC.Publisher.CorporateName%">
<META NAME="DC.Date.Accepted" SCHEME="W3CDTF"
CONTENT="%name=DC.Date.Accepted&meta_table=Dissertation%">
<META NAME="DC.Subject" SCHEME="DDC-Sachgruppe"
CONTENT="%name=DC.Subject&scheme=DDC%">
<META NAME="DC.Subject" SCHEME="freetext" LANG="ger"
CONTENT="%name=DC.Subject&scheme=ger%">
<META NAME="DC.Subject" SCHEME="freetext" LANG="eng"
CONTENT="%name=DC.Subject&scheme=eng%">
<META NAME="DC.Description" SCHEME="freetext" LANG="ger"
CONTENT="%name=DC.Description.Abstract&where [Lang]=ger%">
<META NAME="DC.Description" SCHEME="freetext" LANG="eng"
CONTENT="%name=DC.Description.Abstract&where [Lang]=eng%">
<META NAME="DC.Identifier" SCHEME="URL"
CONTENT="%name=DC.Identifier&scheme=URL&where [DC_Format]=pdf%">
<META NAME="DC.Identifier" SCHEME="URN"
CONTENT="%name=DC.Identifier&scheme=URL&query_field=urn&where [DC_Format]=pd
f%">
<META NAME="DC.Server" CONTENT="Humboldt-Universit&auml;t zu Berlin,
Universit&auml;tsbibliothek">

```

Textstellen, an denen Werte eingesetzt werden sollen, werden mit % gekennzeichnet. Der mit % eingerahmte String ist an das Format, das bei der Übergabe von Werten über die URL benutzt wird, angelehnt, also

*%variablenname=variablenwert&variablenname=variablenwert&....%.*

## Verwendung der Mailtemplate-Variablen

Variable	Bedeutung
name	Name des DC_Elementes aus der Tabelle metadb.DC_Element
meta_table	Name der Spezialisierungstabelle, wenn eine solche zur Identifikation des Elements benötigt wird, aus der Tabelle metadb.DC_Element

scheme	Schema aus der Tabelle metadb.DC_Element
where[]	Array von Feldnamen als Schlüssel und Suchwerten, um die aus metadb.DC_Element gewonnene Query einzuschränken
query_field	Feld, welches zur Ermittlung des Wertes benutzt werden soll, welcher statt des in der Query aus metadb.DC_Element verwendeten benutzt wird

#### 1.5.3.3 Benutzte Klassen

- /mail/Mail.class.php
- /style/MailStyle.class.php
- /dc/DCGenerator.class.php

#### 1.5.4 Der Commit Style

Dieser Style dient zum Erstellen einer Eingabemaske, über die neu angelegte Datensätze in einer bestimmten Reihenfolge in die Datenbank geschrieben werden können. Die entsprechenden Tabellen sind unter dem Element insert\_tables aufzuführen.

*Graphic 5:Der Commit*



edoc - Metadaten: Test

Logout
Neu
Hilfe

File
Resource
Titel
Autor
Beteiligte
URL
Abstract
Keywords
Commit
Suche und Import

Aktion: Datensatz editieren mit Id:  
Titel: titel

Commit

Listen aller editierten Datensätze:
< vorherige |

Resource

Publikationstyp (Test)	Dokument-ID	ALEPH-ID	Rechte	Status	Antragsdatum	Datum der letzten Prüfung	Datum der Veröffentlichung	Sprache	Einrichtung	Aktion																		
Titel																												
<table> <thead> <tr> <th>Titel</th> <th>Sprache</th> <th>Unterarten des Titels</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td>titel</td> <td>Deutsch</td> <td>Titel</td> <td>BEARBEITEN LÖSCHEN</td> </tr> <tr> <td>titel</td> <td>Deutsch</td> <td>Titel</td> <td>BEARBEITEN LÖSCHEN</td> </tr> </tbody> </table>											Titel	Sprache	Unterarten des Titels	Aktion	titel	Deutsch	Titel	BEARBEITEN LÖSCHEN	titel	Deutsch	Titel	BEARBEITEN LÖSCHEN						
Titel	Sprache	Unterarten des Titels	Aktion																									
titel	Deutsch	Titel	BEARBEITEN LÖSCHEN																									
titel	Deutsch	Titel	BEARBEITEN LÖSCHEN																									
Autor																												
<table> <thead> <tr> <th>Akademischer Titel</th> <th>Vorname</th> <th>Nachname</th> <th>Fachbereich</th> <th>Adresse</th> <th>Geburtsdatum</th> <th>Geburtsort</th> <th>Email-Adresse</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td colspan="9"></td> </tr> </tbody> </table>											Akademischer Titel	Vorname	Nachname	Fachbereich	Adresse	Geburtsdatum	Geburtsort	Email-Adresse	Aktion									
Akademischer Titel	Vorname	Nachname	Fachbereich	Adresse	Geburtsdatum	Geburtsort	Email-Adresse	Aktion																				
Beteiligte																												
<table> <thead> <tr> <th>Art der Beteiligung</th> <th>Akademischer Titel</th> <th>Vorname</th> <th>Nachname</th> <th>Fachbereich</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td colspan="6"></td> </tr> </tbody> </table>											Art der Beteiligung	Akademischer Titel	Vorname	Nachname	Fachbereich	Aktion												
Art der Beteiligung	Akademischer Titel	Vorname	Nachname	Fachbereich	Aktion																							
URL																												
<table> <thead> <tr> <th>URL-Schema</th> <th>URL-Adresse</th> <th>Dateiformat (html/xml/pdf usw.)</th> <th>Seitenanzahl</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td colspan="5"></td> </tr> </tbody> </table>											URL-Schema	URL-Adresse	Dateiformat (html/xml/pdf usw.)	Seitenanzahl	Aktion													
URL-Schema	URL-Adresse	Dateiformat (html/xml/pdf usw.)	Seitenanzahl	Aktion																								
Abstract																												
<table> <thead> <tr> <th>Typ</th> <th>Text der Zusammenfassung</th> <th>Sprache</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td colspan="4"></td> </tr> </tbody> </table>											Typ	Text der Zusammenfassung	Sprache	Aktion														
Typ	Text der Zusammenfassung	Sprache	Aktion																									
Keywords																												
<table> <thead> <tr> <th>Klassifikation</th> <th>Schlüsselwörter</th> <th>Aktion</th> </tr> </thead> <tbody> <tr> <td colspan="3"></td> </tr> </tbody> </table>											Klassifikation	Schlüsselwörter	Aktion															
Klassifikation	Schlüsselwörter	Aktion																										

Style

### 1.5.4.1 DTD für Style

```

<!ELEMENT commit (insert_tables, page_ref*)>
<!ELEMENT page_ref EMPTY>
<!ATTLIST page_ref
page_id IDREF #REQUIRED>
<!ELEMENT insert_tables (table+)>

```

#### Commit Style - XML-Elemente

Element	Attribut	Bedeutung
commit		Container für commit Konfiguration
insert_tables		Enthält die Tabellen
table		Identifiziert die Datenbanktabelle

page_ref		Referenz auf eine in der Seitenkonfiguration enthaltene Formularseite für die Anzeige der entsprechenden Liste.
	page_id	id der Seite wie in page.id angegeben

#### 1.5.4.2 Beispiel

```

<page id="i" name="Commit" label="Commit" prev="h"
help_file="../../help/commit.hlp">
<commit>
<insert_tables>
<table name="Person" level="0"/>
<!-- zuerst alle Tabellen die add haben -->
<!-- -->
<table name="C" level="0"/>
<table name="Keyword" level="0"/>
<table name="Resource" level="1"/>
<!-- dann alle Tabellen mit import, -->
<!-- da die ermittelte id in die Reihen aus den aus den vorher
hinzugefuegten -->
<!-- erfolgt -->
<table name="DC_Description" level="2"/>
<table name="Creator" level="2"/>
<!-- after source target -->
<!-- tables have been handled insert into relation tables -->
<table name="DC_Subject" level="2"/>
<table name="Contributor" level="2"/>
<table name="Dissertation" level="2"/>
<!-- dann -->
<!-- alle die warten mussten dass ihre abhaengige id ermittelt worden ist
-->
<table name="Location" level="2"/>
<table name="Title" level="2"/>
</insert_tables>
<page_ref page_id="b"/>
<page_ref page_id="c"/>
<page_ref page_id="d"/>
<page_ref page_id="e"/>
<page_ref page_id="f"/>
<page_ref page_id="g"/>
<page_ref page_id="h"/>
<!--page_ref page_id="i"/>
<page_ref page_id="j"/-->
</commit>
</page>

```

#### 1.5.4.3 Benutzte Klassen

- /metain-bin/styles/CommitStyle.class.php
- /metain-bin/styles/structures/TableOrder.class.php
- /templates\_ger/commit-display.inc.php

## 1.5.5 Der Multi\_in Style

Dieser Style erlaubt die Eingabe mehrerer Datensätze auf einer Formularseite. Editieren und Löschen der Datensätze ist möglich.

Graphic 6: Der multi\_in

Style

### 1.5.5.1 DTD für Style

```
<!-- multi_in -->
<!ELEMENT multi_in
(text|textarea|option|option_by_select|option_by_import|import_by_search|fi
le_import|parameter|dependend|date)*>
<!ATTLIST multi_in
constraint CDATA #REQUIRED
><!-- im Augenblick als constraint nur "no_edit", oder "no_remove" oder
"no_edit|no_remove" um bearbeiten button zu unterdrücken -->
<!-- form fields -->

<!ELEMENT date (value?)>
<!ATTLIST date
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
hidden (yes|no) "no"
required (yes|no) "no"
readonly (yes|no) "no">

<!ELEMENT text (value?)>
<!ATTLIST text
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
hidden (yes|no) "no"
required (yes|no) "no"
```

```

readonly (yes|no) "no">

<!ELEMENT textarea (value?)>
<!ATTLIST textarea
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
required (yes|no) "no"
readonly (yes|no) "no"
parseable (yes|no) "no">

<!ELEMENT option_by_select (select)>
<!ATTLIST option_by_select
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
multiple (yes|no) "no">

<!ELEMENT option_by_import EMPTY>
<!ATTLIST option_by_import
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
searchable (yes|no) "no"
multiple (yes|no) "no">

<!ELEMENT option (value*)>
<!ATTLIST option
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED
multiple (yes|no) "no">

<!ELEMENT parameter (value*)>
<!ATTLIST parameter
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED>

<!ELEMENT dependend (dependend_value*)>
<!ATTLIST dependend
table CDATA #REQUIRED
field CDATA #REQUIRED
label CDATA #IMPLIED>

```

### Multi\_in Style - XML-Elemente

Element	Attribut	Bedeutung
multi_in		Container für Multi_in Style Konfiguration

	constraint	<b>Einschränkungen der Funktionalität des Styles</b> <ul style="list-style-type: none"> <li>• no_edit: aus dem Formular übernommenen Datensätze können nicht wieder editiert werden (nur gelöscht)</li> <li>• no_remove: Datensätze können nicht gelöscht werden (nur editiert),</li> <li>• no_edit no_remove: weder löschen noch editieren</li> </ul>
--	------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 1.5.5.2 Beispiel

```
<page id="c" name="Titel" label="Titel eingeben" prev="b" next="d">
<multi_in>
<!--text table="Title" field="Resource_ID" label="Dokument-ID"
readonly="yes"/-->
<textarea table="Title" field="Title" label="Titel" required="yes"/>
<option_by_select table="Title" field="Lang" label="Sprache"
required="yes">
<select id_field="lang_label">SELECT lang_german, lang_label FROM
Language</select>
</option_by_select>
<option table="Title" field="Qualifier" label="Unterarten des Titels">
<value value="DC.Title" label="Titel"/>
<value value="Title.Subtitle" label="Untertitel"/>
<value value="DC.Title.Alternative" label="Titel alternativ"/>
<value value="DC.Title.Translated" label="Titel übersetzt"/>
<value value="Title.Subtitle.Translated" label="Untertitel übersetzt"/>
<value value="DC.Title.Alternative.Translated" label="Titel alternativ
übersetzt"/>
</option>
</multi_in>
</page>
```

### 1.5.5.3 Benutzte Klassen

- /metain-bin/styles/MultiInStyle.class.php
- /templates\_ger/multi\_in\_display.inc.php

### 1.5.6 Der Multi\_in\_to\_db Style

Erlaubt wie der Multi\_in Style die Eingabe mehrerer Datensätze, es bedarf aber keines Commmit Styles. Die Datensätze können direkt aus der Liste in die Datenbank geschrieben werden. Dieser Style ist im Fall "flacher" Datensätze zu verwenden, die als ganzes in einem Formular eingegeben werden können.

*Graphic 7: Der multi\_in*

Style

### 1.5.6.1 DTD für Style

```
<!-- multi_in_to_database -->
<!ELEMENT multi_in_to_db
(insert_tables, (text|textarea|option|option_by_select|option_by_import|import_by_search|file_import|parameter|dependend|date) *) >

<!ELEMENT insert_tables (table+)>
```

multi\_in\_to\_db Style - XML-Elemente

Element	Attribut	Bedeutung
multi_in_to_db		Container für Multi_in_to_db Style
	insert_tables	Siehe Commit Style

### 1.5.6.2 multi\_in\_to\_db Beispiel

```
<page id="ba" name="DC zuordnen" label="Type zu DC-Elementen zuordnen">
<multi_in_to_db>
<insert_tables>
<table name="DC_for_Type" level="0"/>
</insert_tables>
<option_by_select table="DC_for_Type" field="Type_ID"
label="Publikationstyp">
<select id_field="Type_ID">select Type_ID, Name+'(' +DC_Type+')' from
Type order by Name</select>
```

```

</option_by_select>
<option_by_select table="DC_for_Type" field="Element_ID" label="Element
(name,schema,table to retrieve from)" multiple="yes">
<select id_field="Element_ID">select Element_ID,
Element_Name+", "+Element_Schema+", "+Meta_Table from DC_Element order
by Element_Name</select>
</option_by_select>
</multi_in_to_db>
</page>

```

### 1.5.6.3 Benutzte Klassen

- /metain-bin/styles/MultiInStyle.class.php
- /templates\_ger/multi\_in\_display.inc.php

## 1.5.7 Der Single\_in Style

Dieser Style wird verwendet, um genau einen Datensatz einzugeben. Es wird keine Liste angezeigt, das Formular wird immer mit den jeweils aktuellen Werten gefüllt. Die Übernahme in die Datenbank muss über einen Commit Style erfolgen.

### 1.5.7.1 DTD für Style

```

<!ELEMENT single_in
(text|textarea|option|option_by_select|option_by_import|import_by_search|fi
le_import|parameter|dependend|date)>

```

#### single\_in Style - XML-Elemente

Element	Attribut	Bedeutung
single_in		Container für Single_in Style (analog Single_in Style)

### 1.5.7.2 Beispiel

```

<page id="b" name="Resource" label="allgemeine Dokumentinformationen
eingeben" prev="a" next="c" help_file="../../help/resource.hlp">
<single_in>
<text table="Resource" field="Type_ID" label="Publikationstyp (Test)"
hidden="yes">
<value value="122"/>
</text>
<text table="Resource" field="Resource_ID" label="Dokument-ID"
readonly="yes"/>
<text table="Resource" field="Imported_ID" label="ALEPH-ID"
readonly="yes"/>
<text table="Resource" field="DC_Rights" label="Rechte"/>
<option table="Resource" field="MD_State" label="Status">
<value value="p" label="veröffentlichen"/>
<value value="i" label="zurückstellen"/>
<value value="d" label="löschen"/>
</option>
<date table="Dissertation" field="DC_Date_Submitted" label="Antragsdatum"/>
<date table="Dissertation" field="DC_Date_Accepted" label="Datum der
letzten Prüfung"/>
<date table="Resource" field="DC_Date_Issued" label="Datum der
Veröffentlichung"/>

```

```

<!--text table="Dissertation" field="DC_Date_Created" label="???"
required="no"/-->
<option table="Resource" field="Lang" label="Sprache">
<value value="ger" label="Deutsch"/>
<value value="eng" label="Englisch"/>
<value value="fre" label="Französisch"/>
<value value="lat" label="Latein"/>
<value value="por" label="Portugisisch"/>
<value value="rus" label="Russisch"/>
<value value="spa" label="Spanisch"/>
</option>
<option table="Corporation" field="CorporateName" label="Einrichtung">
<value value="1000000" label="Humboldt-Universität zu Berlin"/>
<value value="1001000" label="Juristische Fakultät"/>
<value value="1002000" label="Landwirtschaftlich-Gärtnerische Fakultät"/>
</option>
</single_in>
</page>

```

### 1.5.7.3 Benutzte Klassen

- /metain-bin/styles/SingleInStyle.class.php
- /templates\_ger/single\_in\_display.inc.php

## 1.5.8 Der File\_import\_page Style

Ein Formular zum Importieren von Dateien wird angezeigt.

### 1.5.8.1 DTD für Style

```

<!-- file_import_page -->
<!ELEMENT file_import_page (preset_field*)>
<!ELEMENT preset_field EMPTY>
<!ATTLIST preset_field
table CDATA #REQUIRED
field CDATA #REQUIRED
value CDATA #REQUIRED >

```

file\_import\_page Style - XML-Elemente

Element	Attribut	Bedeutung
file_import_page		Container für Konfiguration des File_import_page Styles
preset_field		Werte zu übernehmenderTabellenfelder
	table	Identifizier der Tabellenliste
	field	Name des Feldes
	value	Wert des Feldes

### 1.5.8.2 Beispiel



```

<page id="a" name="File" label="ALEPH-Datei importieren" next="b"
help_file="../../help/file_import.hlp">
<file_import_page>
<preset_field table="Resource" field="Type_ID" value="122"/>
</file_import_page>
</page>

```

### 1.5.8.3 Benutzte Klassen

- /template\_ges/file\_import\_display.inc.php
- /metain-bin/styles/FileImportStyle.class.php
- /metain-bin/manager/FileSignal.class.php
- /metain-bin/UploadFile.class.php
- /metain-bin/php-bin/io/File.class.php
- /metain-bin/fileparser/mrcParser.class.php

## 1.5.9 Der Plain Style

Dieser Style erlaubt das Einbinden eines php-Scriptes oder einer html-Datei.

### 1.5.9.1 DTD für Style

```

<!-- plain -->
<!ELEMENT plain (#PCDATA)>
<!ATTLIST plain
file CDATA #IMPLIED >

```

#### plain Style - XML-Elemente

Element	Attribut	Bedeutung
plain		Container für die Konfiguration des Plain Styles
	file	Pfad zum einzubindenden File (wird mit include_once() eingebunden)

### 1.5.9.2 Beispiel

```

<page id="j" name="DBtest (nur für Entwicklung)"
label="db-Models testen">
<plain file="db_models_test.inc.php"/>
</page>

```

### 1.5.9.3 Benutzte Klasse

- /metain-bin/PlainStyle.class.php

## 1.6 Anwendungsdesign

Die einzelnen Dateien sind ausführlich mit Kommentaren versehen und übersichtlich strukturiert, die Quellcodedateien dienen als die wichtigste Quelle für Informationen

*Graphic 8: class structure overview for MetaIn*

### 1.6.1 Der Datenbank-Prozess

Die Tabellen der Datenbank haben untereinander Verbindungen in Form von Foreign Keys. Es gibt abhängige und unabhängige Tabellen in Bezug auf die Reihenfolge des Eintragens. In den Tabellen können mehrere Datensätze auf einmal eingetragen oder editiert werden. Auf Programmdatenebene wird für jede Tabelle eine Datenstruktur erzeugt, die einen Datensatz oder eine Liste von Datensätzen repräsentiert. Das Datenobjekt enthält Datensätze und deren Statusmerkmale und verfügt über Methoden, um aus der Datenbank zu lesen oder zu schreiben.

Felderkonfigurationen werden in `metain-bin/models/FieldType.class.php` gespeichert.

### 1.6.2 Wichtige Funktionsprinzipien

Über den Aufruf von `index.php` im jeweiligen Verzeichnis wird die Anwendung gestartet. Dabei wird die Datei `/metain-bin/main.php` aufgerufen, die die Klasse `/metain-bin/manager/Manager.class.php` initialisiert. Das Managerobjekt wird in der Session zusammen mit allen Attributen gespeichert. Darunter sind auch die Konfigurationen der DB-Struktur (in `/metain-bin/sstructures/dbStructure.class.php`) und der Seiten (`/metain-bin/sstructures/SiteStructure.class.php`). Die XML-Dateien werden mit Hilfe der Klasse `/metain-bin/php-bin/xml/ConfigurationParser.class.php` geparkt. Klassen, die XML-konfigurierbar sein sollen, implementieren eine spezielle XML-Schnittstelle. So kann die Konfiguration bei Weiterentwicklung leicht angepasst und neue Styles hinzugefügt werden. Über die `main`-Methode des Managers wird die `main`-Methode des Styles der aktuellen Seite aufgerufen. Dabei wird das Request-Signal in einem Signal-Objekt aufgearbeitet und dem Style übergeben, wo die eigentliche Funktionalität der Seite liegt. Es gibt eine Reihe unterschiedlicher Signale, die von den unterschiedlichen Styles verwendet werden können und ihre eigene Formatierung bei der HTML-Ausgabe und unterschiedliche Filter mitbringen. Die Datenbankverbindung wird in `$_SESSION["manager"]` (`Manager.class.php`) initialisiert, ist global als `dbConnection` verfügbar und wird von allen Objekten für den DB-Zugriff benutzt.

Das Verzeichnis `/php-bin/` enthält generische Klassen für den allgemeinen Gebrauch, darunter auch `/php-bin/common/common.inc.php`, wo Funktionen für den Import der Klassen definiert werden.

Die Templates sind `php`-Files mit reiner Ausgabelogik, die lokal innerhalb der einzelnen `Display()`-Methoden der Seitestyles aufgerufen werden und deren Methoden benutzen.

Die einzelnen Styles werden über die `/metain-bin/styles/PageStyleFactory.class.php`-Klasse erzeugt und sind von `PageStyle.class.php` abgeleitet.

Die wichtigsten speziellen Klassen für die Listenausgabe und die Formulare befinden sich in `/metain-bin/gui/` und werden über die `/metain-bin/gui/FormFieldStyleFactory.class.php` erzeugt.

Die Klassen, die die Datenbankeinträge aufnehmen, befinden sich in `/metain-bin/models/`, werden beim Lesen der Konfiguration initialisiert und in `dbStructure` verwaltet. Aus der DB-Konfiguration werden sowohl Listen als auch ein Graph für die Tabellenverbindungen (`links`) erzeugt.

## Chapter 3

### MetaOut / MetaSearch

#### Deliverable D1.4 Digital Repository – Germany

# Inhaltsverzeichnis

1	MetaOut.....	3
1.1	Installation.....	3
1.2	Erzeugen neuer Browsing Strukturen .....	3
1.2.1	Installation neuer Browsing Strukturen.....	4
1.2.2	XML-Konfiguration .....	5
1.3	Templates .....	8
1.4	Caching.....	8
1.5	offene Aufgaben .....	8
2	MetaSearch.....	9
2.1	Instalation.....	9
2.2	Konfiguration .....	9
2.2.1	Verzeichnisse erstellen .....	9
2.2.2	Die Datei metasearch.ini .....	9
2.2.3	xml-Konfiguration.....	9
2.3	Anwendungs Design .....	9

# 1 MetaOut

MetaOut ist eine Anwendung zum Abrufen von digitalen Dokumenten über ein Web-Interface. Die Anwendung bietet eine einfache Schnittstelle zwischen dem User und einer SQL Metadatenbank, die die Wiederherstellung von Dokumenten gestattet. Der Quelltext ist in PHP geschrieben. Das Anwendungsdesign basiert auf Objekt Orientierten Prinzipien und dem MVC 2 Muster. MetaOut kann über xml-Datei konfiguriert werden.

Einige Eigenschaften:

- Logging
- XML-Datei Konfiguration
- PEAR (generischer Datenbank Access Layer)
- Caching
- Templates

## 1.1 Installation

Um die Applikation starten zu können brauchen Sie einen PHP fähigen HTTP-Server (z.Bsp.: APACHE). Stellen Sie sicher dass die PHP Erweiterungen für das RDBMS (Sybase) korrekt installiert sind.

## 1.2 Erzeugen neuer Browsing Strukturen

Zum Erzeugen neuer Browsing-Strukturen sind 2 Schritte notwendig:

1. Es muss jeweils ein Verzeichnis für die index.php-Datei, für die gecachten html-Seiten, für log-Dateien und für die spezielle Templates erzeugt werden.
2. Die xml-Konfigurations Datei muss angepasst werden.

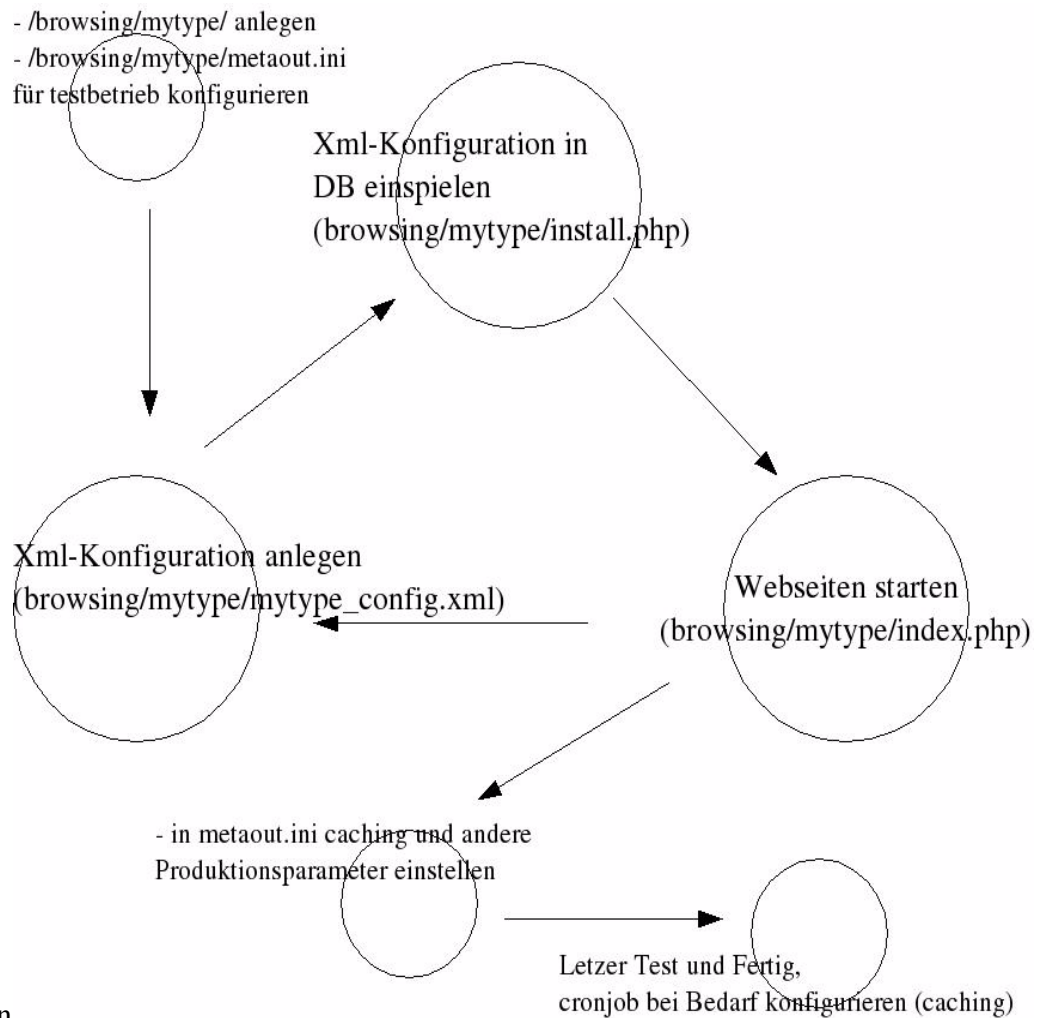
Wenn Sie eine valide Konfigurations Datei erstellt haben können Sie die index.php-Datei über einen Browser, mit dem Parameter 'admin\_action=install' aufrufen. Um den Cache aufzufrischen können Sie den Parmameter 'admin\_action=recache' nutzen.

Aufrufform (URL):

`http://[ihr servername]/[Verzeichnis der index.php-Datei]/index.php?admin_action=install`

Bei diesem Beispielbefehl wird die xml Konfigutation eingelesen und zu einem Baumstruktur kompiliert, die in der Datenbank gespeichert wird. Danach werden die Knoten des Baumes dynamisch aus der Datenbank generiert um die html Seiten aufzubauen.

*Graphic 1:* Installation and Entwicklungs Zyklus für neue Browsing



## Strukturen

### 1.2.1 Installation neuer Browsing Strukturen

Zur Installation neuer Browsing Strukturen ist unter dem root-Verzeichnis ein neues Verzeichnis anzulegen (dieses kann auch an anderer Stelle angelegt werden, aber dann ist die Variable \$myPath und \$gfPath in index.php anzupassen ). Dieses Verzeichnis sollte folgende Dateien enthalten, die man am besten aus laufenden Konfigurationen kopiert und dann anpasst:

Dateien für den Aufruf der Browsing Struktur

Dateiname	Beschreibung
index.php	Startpunkt für das Browsing
metaout.ini.php	Konfigurationsparameter (Datenbank, Logging, Caching, Pfade etc. ,nähere Beschreibung in der Datei selbst)
site_structure_browsing_[name].xml	Xml-Konfigurationsdatei, Name der Datei kann in metaout.ini geändert werden

## 1.2.2 XML-Konfiguration

Um die Konfiguration vornehmen zu können brauchen Sie gute Kenntnisse in xml und SQL. Die dtd befindet sich im Modul MetaOut unter dtd/metaout.dtd.

Das Browsing hat eine baumartige Grundstruktur. Der User navigiert im Baum, wobei die Anzeige sukzessive verfeinert wird. Der Baum besteht aus Wurzel, Knoten und Blättern (Endelemente). Ein Knoten hat genau einen parent-Knoten, kann mehrere child-Knoten besitzen, dadurch ergibt sich ein eindeutiger weg von der Baumwurzel zu jedem Blatt. Der eindeutige Weg von der Wurzel zu einem Knoten oder Blatt wird als Pfad des Knotens bezeichnet. Die Knoten mit gemeinsamem parent-Knoten werden als Nachbarn bezeichnet. Der Baum wird grafisch so dargestellt, dass die Wurzel oben liegt und die Blätter unten. Knoten gleicher Ebenen werden horizontal angeordnet.

Die Baumstruktur ist über eine xml-Datei konfigurierbar. Der Browsing-Baum ist ein dynamischer Baum, einige Knoten stehen mit der Konfiguration fest (statisch) andere werden dynamisch mit Hilfe von select-Statements aus der Datenbank generiert. Die Konfiguration umfasst eine Sammlung von Regeln, um diesen dynamischen Baum zu erzeugen. Die Dynamik wird dabei über sich verfeinernde SELECT-Anfragen erzeugt (Akkumulation von Bedingungen in der where-Klausel). Jedes Element in der xml-Konfiguration beinhaltet entweder die Bauanleitung zu einem statischen Knoten oder die Vorschrift zur Erzeugung einer Anzahl von Nachbarknoten. Statische und dynamische Elemente können dabei sowohl horizontal als auch vertikal gemischt werden.

In der html-Ausgabe wird dem User einen Ausschnitt der Baumstruktur angezeigt. Das Label des aktuellen Knoten und seiner Nachbarelemente erscheinen links in der Navigation, wobei der aktuelle markiert ist. Die Label der child-Knoten, des aktuellen Knotens, erscheinen rechts und können angeklickt werden, um so zur nächst tieferen Baumebene zu gelangen. Erreicht man so die Blätter des Baumes (\_end) wird rechts eine Detailübersicht angezeigt (final\_), die die Zielelemente des Browsers darstellt (bspw. Dokumente). Innerhalb dieser Zielelemente kann mit Hilfe eines lokalen Pagings geblättert werden. Oben sieht der User den Pfad zum aktuellen Knoten. Baumebenen können während der Browsers übersprungen werden, dies ist in der Konfiguration festgelegt oder wird dynamisch ermittelt, wenn es nur ein child-Element gibt.

Der schwierigste Teil des Konfiguration besteht im Zusammenbauen komplexer SQL-Abfragen mit mehreren Joins. Testen Sie ihre Anfragen bevor Sie sie in die Konfiguration übernehmen.

Mit Hilfe des SELECT-Statements (mit " as varname" ) werden Variablenwerte aus der Datenbank erzeugt, die teilweise festgelegt sind, z.B. alle Variablen mit Prefix "page\_", oder neu angelegt werden. Die Variablen sind dann in Templates als \$GLOBALS["viewerBean"]->stepElements[\$index]->conditions['varname'] zugänglich. Wichtiger ist aber, dass diese nicht vorbelegten Variablennamen in den unteren Ebenen sichtbar werden, um dort in der where-Klausel die Auswahlmengen festzulegen, z.B. "where Resource\_ID=\$varname ", diese conditions werden in der GET-Variablen c[Knotenid][varname]=varwert weitergereicht.

### Bedeutung der xml-Elemente

Element	Attribut	Bedeutung
Browsing		root-Element der Konfiguration
step_static		Konfiguration eines statischen Knotens

	page_label	Anzeigename in Navigation zu diesem Knoten
	show_style	Name des Templates das verwendet werden soll, der Wert von show_style ergänzt um ".inc.php" liefert den Namen der Template-Datei im Template-Verzeichnis (wird in metaout.ini.php) angegeben
step_select		Konfiguration eines dynamischen Knotens oder besser einer Menge zur Laufzeit erzeugter Knoten
	show_style, page_label wie bei step_static	page_label wird eventuel von select_query überschrieben falls dort " as page_label" auftaucht
	select_query	SELECT-statement, die mit select gelesenen Spaltennamen der Resultatmenge können mit Namen belegt werden, bspw. "select Title as page_label, ... ", diese werden evntl. vom Template oder von den StepElement-Klassen benutzt (page_label, page_item_proprintable,...) und namentlich vorgegeben, oder für die Verwendung in Abfragen auf Unterebenen benötigt, dann sind sie frei wählbar (für Sybase weniger als 30 Zeichen)
	jump_one_step	"0" (default) oder "1", um in der Anzeige eine Ebene zu überspringen und damit zum ersten Navigationspunkt der dann folgenden Ebene zu gelangen
	local_header_query (wie bei step_final)	Möglichkeit extra Daten ins Template zu bringen (meistens für Überschrift verwendet)
step_select_end		Blätter im Baum die dynamisch generiert werden sollen, muss ein step_final enthalten. Sonst wie step_select
step_static_end		Ein statisches Blatt. Muss Step_final enthalten. Sonst wie step_static
step_final		Anzeige der Endseite des Browsings, kann eine Liste sein oder ein Inhaltsverzeichnis oder ..., dies hängt alles vom Template ab
	show_style	template name



select_query	wie bei select_step, variable page_item_id wird von den meisten Templates als die eindeutige Id des Datensatzes in Zusammenhang mit config/filters/FormatFilter.class.php verwendet, um bspw. den ´ Link für die servereindeutige Frontpage für ein Dokument zu generieren.
query_implode_col	Query implode column - Aggregieren über gewisse Spalten der von select_query gelieferten Resultate, der Algorithmus ist so - es werden die Elemente zusammengefasst die sich nur in query_implode_col unterscheiden, es wird nicht mit Id gearbeitet, da man vielleicht nicht immer eine hat. Im Programmcode : FinalPathElement.fetchItems(\$sql, \$from=0, \$to = null )
page_count_query	ein SQL-Query der Form "select count(*) from Resource R join Relation ..." o.ä., um die Anzahl der Datensätze, die für die Ausgabe gefunden werden zu ermitteln, wegen paging ist dies als extra-Abfrag erforderlich.
page_navigation_style	Die Art wie in der Endausgabe lokal navigiert wird: count_paging - nach Anzahl der Datensätze navigieren letter_paging - paging nach Symbolen wie Anfangsbuchstaben (bedarf einer extra query siehe unten) list_all - es werden alle Datensätze aufgelistet, kein Paging
page_navigation_query	SQL-Abfrage für die Navigationssymbole - Bsp: page_navigation_query="select distinct upper(substring(P.PersonalName_L,1,1)) as page_letter from Resource R join Relation RL on R.Resource_ID=RL.Resource_ID join Title T on R.Resource_ID=T.Resource_ID left outer join Creator C on C.Resource_ID=R.Resource_ID join Person P on C.Person_ID=P.Person_ID where RL.Object_ID=\$journal_id and RL.Type='IsPartOf' and T.Qualifier='DC.Title' "
local_header_query	SQL-Abfrage um zusätzliche Daten in das Template zu bringen , kann benutzt werden um Überschriften zu generieren, die als " as varname" gesetzten Variablen sind im template als \$GLOBALS["viewerBean"]->local_header['varname'] erreichbar

Die wichtigsten Klassen für die Konfiguration von Elementen sind

manager/ConfigurationManager.class.php, manager/PathElement.class.php (für Knoten) und manager/FinalPathElement.class.php (für Blätter). Der angezeigte Pfad wird mit manager/Path.class.php erzeugt.

## 1.3 Templates

Das Template System ist zur Zeit nicht vollständig implementiert. Es gibt ein zentrales Template (index.tpl), welches mithilfe von der XIPE Syntax geschrieben wurde (XIPe ist ein PHP Template System, zu erhalten als PEAR Packet). Die dynamisch enthaltenen Teile der einzelnen Seiten enthalten Presentations Logik, geschrieben in PHP.

Templates für die HTML Darstellung

Template	Beschreibung
index.tpl	Die gesamte Ausgabe ist von hier koordiniert. Der xml Header wird für die Konfiguration des Template Engine benutzt.

## 1.4 Caching

Es wird ein Parameter abhängiger Caching Mechanismus des Templates XIPE benutzt. Über das Attribut `cach_time= "Wert"` (0 = ewig, "" = kein Cache) kann im xml Header der index.tpl-Datei die Caching Zeit eingestellt werden. Um den Cache zu erneuern sollten Sie in der Nacht einen Cron Job laufen lassen.

Zwei Faktoren sind wichtig:

- Entfernen Sie die gecachten Dateien aus dem TMP Verzeichnis.
- Führen Sie einen HTTP Agenten (Webget, Lymx) aus, um neue Cache Dateien zu erstellen.

Das Skript, welches der Cron Job aufruft ist:

```
# crontab -l
```

```
-> /usr/local/www/bin/helper/clean_cache.sh
```

## 1.5 offene Aufgaben

- Überführen persistenter URLs mit Kompilieren der Baum in die Datenbank
- Intelligentes Paging, in Hinsicht auf UTF-8
- Level-abhängiges Caching.
- Sicherheit
- Template Sprache
- CSS-basierendes Design
- DTD für die xml Konfigurationsdateien

## 2 MetaSearch

MetaSearch ist ein xml konfigurierbares Tool um in einer SQL-Datenbank zu suchen.

### 2.1 Instalation

Der Quelltext läuft unter PHP 4.3.8. Es wurde mit Sybase getestet und für MYSQL vorbereitet. Kopieren Sie den Code in ein Web Verzeichnis und verändern Sie die global\_settings.inc-Datei. Folgen Sie den Instruktionen in der Konfiguration. Beispiel Templates sind in /templates\_ger zu finden.

### 2.2 Konfiguration

#### 2.2.1 Verzeichnisse erstellen

Erstellen Sie ein Verzeichnis für jede spezielle Such Form (/standard) und kopieren Sie die Dateien index.php, metasearch.ini und standard.xml hinein (Wenn Sie die .xml-Datei umbenennen wollen müssen Sie Änderungen in der metasearch.ini-Datei vornehmen.)

#### 2.2.2 Die Datei metasearch.ini

Die Datei metasearch.ini sollte im selben Verzeichnis liegen wie die index.php-Datei. Konfigurieren Sie den Debugging Modus, den Datenbank Zugang, den Pfad zu den Template Dateien und den Dateinamen für die xml Konfiguration.

#### 2.2.3 xml-Konfiguration

Attribut Werte für den search\_mode

Attribut	Bedeutung
macht_part	like '% word%'
match_part_begin	Like 'word%'
match_whole (default)	= 'word'
match_integer	= value

### 2.3 Anwendungs Design

Das Design folgt in etwas dem MVC 2 Pattern. Die UML (.xmi)-Datei befindet sich im Repository MetaOutDocs/figs/.

*Graphic 1:* Klassen Struktur für MetaSearch (UML-Klassendiagramm)

