Project Documentation

Diploma project at TIK, ETH Zurich by Alexander Karg
Supervision by Dr. Erik Wilde and Prof. Dr. Bernhard Plattner
Technology provided by XLinkbase

Index

Project Documentation 1
Index 2
1 Introduction 3
1.1 Picture Archive 3
1.2 This documentation 3
2 Project management 3
2.1 Project goals 3
2.2 Picture Archive ver.0.1 5
2.3 Picture Archive ver.0.2 5
3 Implementation 6
3.1 Programming language: JAVA 6
3.2 Architecture 6
3.3 Imported components and modules 7
3.4 System requirements 7
3.5 Installer 7
3.6 Files/Classes 8
3.7 Data model 9
3.8 Export 10
4 Outlook 11
4.1 Picture Archive ver.1.0 11
4.2 Other directions of development 12
Appenix A: dig35 definitions 13
Appenix B: Glossary 13
Appenix C: Picture Archive ver.0.2 - Issues 14
Appenix D: Project Goals (Aufgabenstellung) 15

This domcument is written for 'Picture Archive' ver.0.2.
Revision 0 of this document, August 18, 2001

1 Introduction

1.1 Picture Archive

'Picture Archive' is a program for cataloging, browsing and managing your image files. 'Picture Archive' offers a specialised data model for image files, which enables to manage an extraordinary depth of metadata, i.e. information about your pictures. It offers an 'export to HTML' function, so you can share your photos with others.

Notes on the latest Picture Archive ver.0.2:
If you experience any problems when starting the program by executing PICARC.EXE, do run PICARC.BAT instead. Make sure you have installed the necessary JRE (1.3.1 or later) installed, before running PICARC.BAT.

1.2 This documentation

This project documentation describes the process how the 'Picture Archive' program has been designed, gives an overview of the software architecture and an outlook of applications that go beyond the implementation scope of this diploma project. For information on how to use the 'Picture Archive' GUI please see the 'Picture Archive, User Guide'.
This documentation does not give any implementation details, it is therefore not suited to give anyone the information required to expand, modify or debug the program code.

2 Project management

2.1 Project goals

The formal project goals ('Aufgabenstellung') can be found in "Appenix D: Project Goals (Aufgabenstellung)".
Ongoing from the formal goals the following goals have been listed and prioritised:

Basic Functions (priority: 1, 1.Phase)

- GUI, with following properties sorted in order of relevance:
  - Edit, Add, Delete Topics, Relations from XLB db.
  - Fast to Use (for accustomed users), click count, templates
  - Ease of Use (for novice users)
  - Stability
  - Performance
- design and implement regarding the further phases
- English language version
- auto thumbnails
- jpeg image format supported

Extended Functions (2. Phase)

- Priority: 2
  - gif image format support
  - queries
  - support text files as external resources
  - support html files as external resources
  - search function within Java Application
  - file and directory management functions
  - customisable layout for html output
  - image properties auto detection
  - i18n: German language version
  - Installer for Win32

- Priority: 3
  - customisable java interface
  - all common image formats
  - Slideshow (Java and/or HTML)
  - Camera and Scanner Acquisition
  - customisable sorting for GUI and/or output
  - Tutorial
  - usability test with novice users
  - minimal configuration tests
  - performance tests
  - reliability tests
  - browsing/selection of images/text files on hard drive
  - browsing/selection of images/text files from www
  - import for misc. other picture management tools

- Unprioritized
  - Multiple Image Windows
  - Zoom, with 'Shrink to Fit' and 'Stretch to Fit'
  - templates
  - import for Kodak photo cd
  - search/index for time data
  - html preview in application
  - GUI according to Sun Java GUI guidelines
  - Java Code according to Sun Java Coding guidelines
  - Installer for Solaris Unix
  - Installer for Mac

Functions of priority 1, are defined to be basic functions and are considered as minimal requirement for the program. Therefore it has been decided to split the project into two phases. In the first phase all basic functions got implemented and tested. In the second phase, which was planned to last until 26 July 2001 as many as possible of the remaining functions got implemented.

The following task and dates have been defined:

Task	**Duration** (days)	**Start**	**End**
Phase 1: Basic Functions		Do 26.04.01	Mo 18.06.01
Analysis, Planning, Design Java Classes	14	Do 26.04.01	Fr 11.05.01
Implementation Java GUI, Java Classes	22	Mo 14.05.01	Di 05.06.01
Modifications XSLT für HTML output	10	Mo 14.05.01	Fr 25.05.01
Unit Test, Modifications	9	Mi 06.06.01	Mo 18.06.01
Phase 2: Extended Functions		Di 19.06.01	Do 26.07.01
Analysis, Planing, Design, Implementation	21	Di 19.06.01	Di 17.07.01
Unit Test	7	Mi 18.07.01	Do 26.07.01
System Test	5	Mo 06.08.01	Fr 10.08.01
Documentation	6	Mo 06.08.01	Mo 13.08.01
Final Touch	4	Di 14.08.01	Fr 17.08.01
Reserve	6	Fr 27.07.01	Fr 03.08.01

2.2 Picture Archive ver.0.1

The first version Picture Archive 0.1 has been built on June 19th 2001. It included all goals of priority 1.

2.3 Picture Archive ver.0.2

The version 0.2 of Picture Archive has been built on August 13th 2001. The following functions have been added since ver.0.1 (all first priority functions):

- Support for gif format
- Query function
- Installer for Win32
- Thread for thumbnail loading
- 'Shrink to Fit'
- Templates -> series
- Thumbnail caching

More implemented features and enhancements are listed in “Appenix C: Picture Archive ver.0.2 - Issues”.

3 Implementation

3.1 Programming language: JAVA

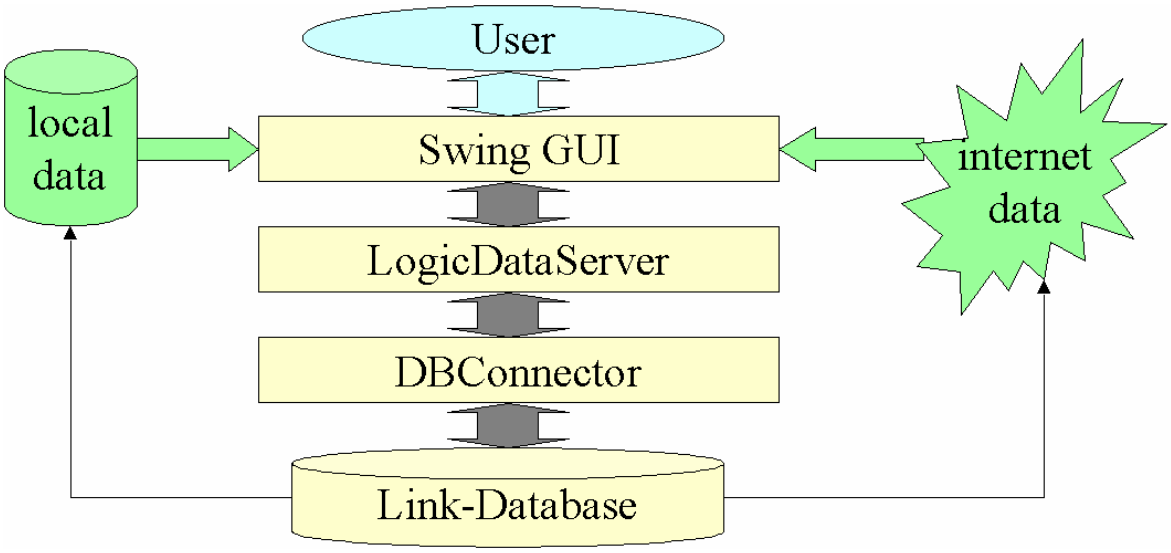
Java has been chosen as Programming Language for the following reasons:

- Platform independency
- Support for XML, XSLT, XPATH
- General good availability for external modules to import
- Java is well extensible through object orientation, and therefore well suited for the programming method, that has been used in the 2. phase

3.2 Architecture

The applications main task is to administrate metadata. This means the data, e.g. the images, are not part of the application, but are considered to be external resources anywhere on the net or on the local hard disk.

The application is logically split into three layers:



All three layers are implemented in Java.

All user interaction is handled by the Swing GUI layer. The GUI layer loads the images from the external location to show them on the screen. All manipulations or requests on the metadata have to be done through the LogicDataServer.

The LogicDataServer layer knows the data model and offers a bundle of functions to the GUI layer, such as ‘Give me all topics that are in XY-relation to the YZ-topic!’. This layers does caching of metadata and of thumbnails.

The DBConnector is responsible for low level database connection, in this case for read and write actions to the Link-Database file.

The Link-Database is a single XML file.

This architecture has the benefit that each layer can be replaced without needing to change the others. Especially the DBConnector and the XML file could be replaced by a (faster) relational database or the GUI could be redesigned reusing the LogicDataServer.

3.3 Imported components and modules

The following external components and modules have been used:

- Xalan-Java 2: xerces.jar, xalan.jar for XML, XSLT, XPATH
- HBWorks: ExtDirpane.jar

3.4 System requirements

Not yet specified.

Notes on the latest Picture Archive ver.0.2:
The performance, and so the system requirements, in this version depends heavily on the number of pictures in the archive.
The program has been tested on a Pentium II with 128 MB RAM on Windows98 and Windows2000. This version requires 80 MB when installed on your hard disk.

3.5 Installer

The Installer has been created by the FreeWare Version 3.5.3 of InstallAnywhere NOW! by Zero G Software (www.zerog.com). The installer creates a self extracting archive with the Java Virtual Machine included.

3.6 Files/Classes

The ver.0.2 built consists of 52 java class files with a total size of 407 KB. The main 'picarc' directory includes the following files and folders:

files:

picarc.exe	the executable (generated by the Installer)
picarc.xml	data file
empty.xml	empty data file (needed for export)
picarc.xslt	XSL transformation file (needed for export to HTML)
picarc.doc	user guide
picarc.jpg	startup image
picarc.bat	alternative startup method (using your local JVM, std_out to window)
options.xml	program options file
lax.jar	installer helper file
picarc.jar	installer helper file

folders:

classes	internal and external classes
cache	empty folder for the cached thumbnails
html	files need to generate HTML output
demo_images	contains the demo images
jre	installer helper files
Uninstaller..	Uninstaller

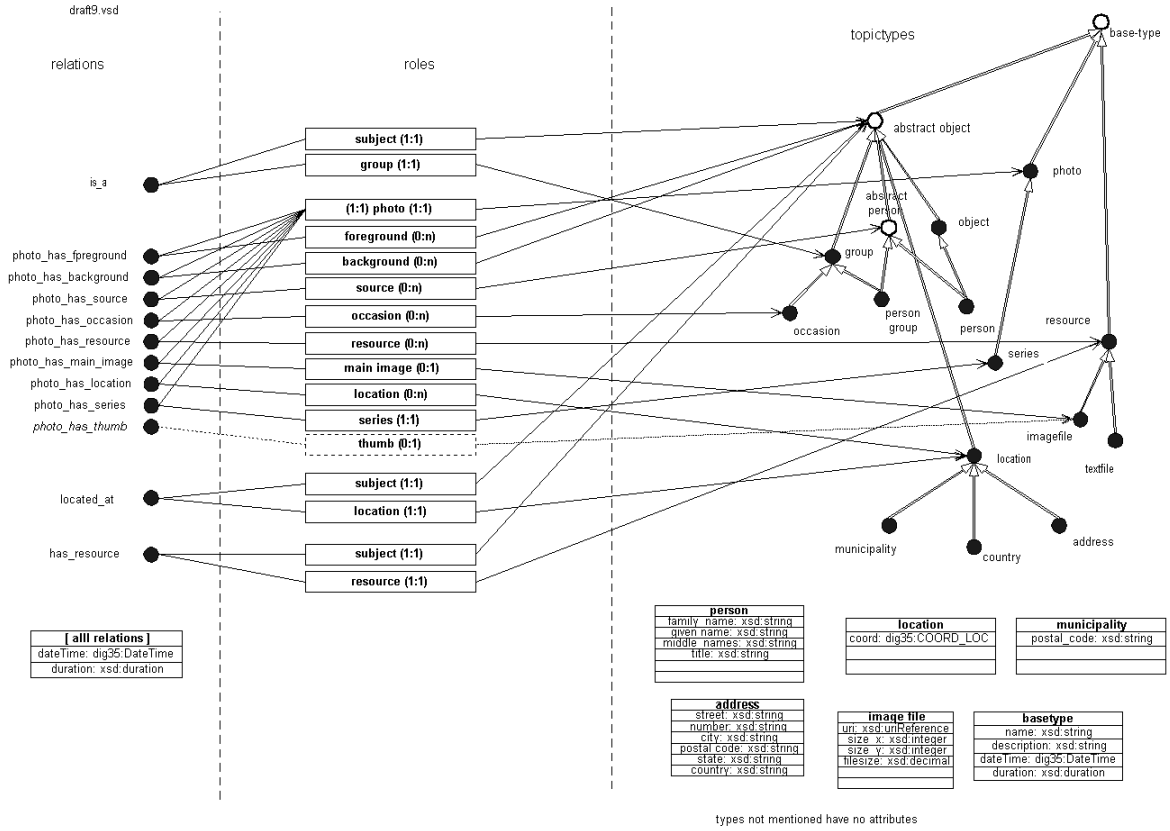
Notes on the latest Picture Archive ver.0.2:
The size of the installer INSTALL.EXE is 49 MB. It does include the JVM, which is about 5 MB. By checking out all unnecessary classes the size of the installer could be reduced to less than 2 MB (without JVM) or less than 7 MB (with JVM).

3.7 Data model

The data model chosen to manage metadata of images is based on the XLinkbase Technology (www.xlinkbase.com) and on a former semester project (XLinkbase Picture Archive).

The model has been simplified to better match the needs of Picture Archive.
There are only binary relations, meaning that each relation has exactly two roles, whose topics must be of the specified type or a subtype of it.

The following schema shows the data model:



Thin arrows connect associations, roles and topictypes. Thick arrows (x --> y) mean x is subtype of y. Topictypes with non-filled circles represent abstract topictypes, meaning that topics cannot be directly of such a type. The boxes show the attributes of a specific type. Subtopics do inherit the attributes of their supertypes.

All attributes are either typed according to XML Schema language (www.w3c.org, xsd:...) or according to (http://www.digitalimaging.org/) DIG35 Specification: Metadata for Digital Images. Version 1.0. See Appendix "Appenix A: dig35 definitions" for detailed definitions of the used dig35 datatypes.

Notes on limitations of the latest implementation (Picture Archive ver.0.2) :
There is no type checking done by the application. The datatypes dig35:COORD_LOC and type dig35:DateTime are treated the same as xsd:string. There is no support for attributes of relations.

A very simple dtd for the data file is:

```
<!ELEMENT picarc (topics, relations)>
<!ELEMENT topics (topic*)>
<!ELEMENT topic (name?, attribute*)>
<!ELEMENT relations (relation*)>
<!ELEMENT relation (role*)>
<!ELEMENT role EMPTY>
<!ATTLIST topic
    type CDATA #REQUIRED
    TID ID #REQUIRED
>
<!ATTLIST relation
    type CDATA #REQUIRED
    RID ID #REQUIRED
>
<!ATTLIST role
    type CDATA #REQUIRED
    TID IDREF #REQUIRED
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT attribute (#PCDATA)>
<!ATTLIST attribute
    name CDATA #REQUIRED
>
```

For example the following is a valid data file:

```
<picarc>
    <topics>
        <topic TID="t_1" type="series">
            <name>Default Series</name>
        </topic>
        <topic TID="t_2" type="object">
            <name>Grossmünster</name>
            <attribute name="description">The biggest church in Zurich</attribute>
        </topic>
        <topic TID="t_3" type="group">
            <name>church</name>
        </topic>
    </topics>
    <relations>
        <relation RID="r_1" type="is_a">
            <role TID="t_3" type="group"/>
            <role TID="t_2" type="subject"/>
        </relation>
    </relations>
</picarc>
```

3.8 Export

'Picture Archive' offers the possibility to export all or a set (query) of photos to XML or HTML. 'Export to HTML' will create a set of HTML pages with thumbnails. The export to HTML function is implemented as follows.

The selected photos get exported to a temporary XML file, including the metadata according to the chosen 'Depth of Metadata' (see user guide for more information). This temporary XML file gets transformed via an XSLT 1.1 stylesheet to a set of html pages. The thumbnails are generated by the LogicDataServer layer.

4 Outlook

4.1 Picture Archive ver.1.0

The following features, enhancements, bugs or processes have not been taken care of yet, but should be regarded towards the implementation of 'Picture Archive ver.1.0':

- Testing on platforms: Win95, Win98, Win2000, WinNT, WinXP, Linux, Solaris, MacOS
- Massive improvement of performance for a higher number (>1000) of photos is necessary, maybe by using a relational database
- Toolbar with history functions
- Enhance support for attributes DateTime and Coordinates
- Add NOT operator to query composer
- Improve User Guide / Tutorial
- Help System: Software Help, Tooltip-help
- Allow to set properties of a photoset(query) by assigning a new series
- Improve error catching and propagation
- Support for external resources, html, textfiles
- Implement all planed options for export
- Customisable layout for html output
- Image properties auto detection
- i18n: other languages
- All common image formats
- Slideshow (Java and/or HTML)
- Usability test with novice users
- Minimal configuration tests
- Performance tests
- Reliability tests
- Browsing/selection of images/text files on hard drive
- Browsing/selection of images/text files from www
- Import for misc. other picture management tools
- Search function within Java Application
- File and directory managment functions
- Drag and Drop from OS or Browser to add images

The list of current issues can be found in "Appenix C: Picture Archive ver.0.2 - Issues"

4.2 Other directions of development

The following features would aim to a complete digital image management system, and therefore step away from the idea that the metadata and the data should be handled separately:

- image conversion
- image retrieving (Camera and Scanner Aquisition, ex. TWAIN)
- image enhancing (colors, filters, turning, etc..)

Another direction one might think of is aiming to a shareable, multiuser picture archive where users can check in and check out images, respectively their metadata, using a common extensible metadata model.

Yet another direction would be using image pattern recognition programs to try to automatically gain meta-information from the pictures.

Appenix A: dig35 definitions

```
<xsd:complexType name="tDateTime">
    <xsd:choice minOccurs="0" maxOccurs="1">
       <xsd:element name="EXACT" type="xsd:timeInstant"/>
       <xsd:element name="DATE" type="xsd:date"/>
       <xsd:sequence>
           <xsd:element name="MONTH" type="tRecurringMonth"
              minOccurs="0" maxOccurs="1"/>
           <xsd:element name="YEAR" type="xsd:year" minOccurs="0" maxOccurs="1"/>
           <xsd:element name="CENTURY" type="xsd:century"
              minOccurs="0" maxOccurs="1"/>
       </xsd:sequence>
    </xsd:choice>
    <xsd:element name="WEEK_DAY" type="dig35:tLangString" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="SEASON" type="dig35:tLangString" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="COMMENT" minOccurs="0" maxOccurs="1"/>
    <!--xsd:attribute ref="TIMESTAMP"/-->
    <!--xsd:attribute ref="xml:lang"/-->
</xsd:complexType>


<xsd:element name="COORD_LOC">
    <xsd:complexType>
       <xsd:element name="LONGITUDE" type="tDegree" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="LATITUDE" type="tHalfDegree" minOccurs="0" maxOccurs="1"/>
       <xsd:element name="ALTITUDE" type="xsd:double" minOccurs="0" maxOccurs="1"/>
       <!--xsd:attribute ref="TIMESTAMP"/-->
    </xsd:complexType>
</xsd:element>
```

Appenix B: Glossary

GUI	Graphical User Interface
HTML	Hyper Text Markup Language
JVM	Java Virtual Machine
XLB	XLinkBase
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Appenix C: Picture Archive ver.0.2 - Issues

(August 17, 2001)

1	Enhancement	Low	QueryTree, make it dragable within tree
2	Enhancement	Low	Enable faster loading of dialogs like addnewphotos
3	Enhancement	Low	add thumb preview in addPicturesFrame or/and in FileLoader
4	Implemented	Medium	export2html: add topicTypes to index
5	Implemented	High	improve picture loading thread
6	Open bug	Medium	active windows, @ new topic, @ new from new relation frame
7	Implemented	High	max. nof pictures per query
8	Enhancement	Low	export _html: info about query, ex.how many photos
9	Trash	Low	-
10	Open bug	High	tab order, and default buttons
11	Feature	Low	toolbar: browse history functions, save
12	Implemented	High	unsaved changes? dialog before quitting
13	Implemented	High	date/time parameters
14	Implemented	High	thumbnail caching
15	Enhancement	Low	index: display all thumbs of types and allow queries
16	Fixed	High	must save before export, because of web/thumbs
17	Open bug	Low	errors in picarc.xml lead to NPE
18	Implemented	High	xml/html output queries
19	Enhancement	Low	html-output: improve, repeating thumbnails eg. persons
20	Implemented	Not Entered	scale index window
21	Implemented	Medium	add option: show pictures in orig size or fit to frame
22	Open bug	Medium	Compatibility for WIN2000
23	Enhancement	Medium	enhance support for attributes DateTime and Coordinates
24	Open bug	Medium	html-export: alphabet. sort of index contents
25	Open bug	Medium	index: alphabet. sorting
26	Feature	Low	support for external resources, html, textfiles
27	Enhancement	Low	use validate and stuff to improve adjustFrames and adjustFrames2
28	Enhancement	Low	allow to set props of a photoset(query) by assigning a new series
29	Open bug	Low	improve error catching and propagation
30	Open bug	Low	changes in edit field, should be saved immediately

Appenix D: Project Goals (Aufgabenstellung)

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérerale de Zurich
Politecnico federale di Zurigo
Institut für
Technische Informatik und
Kommunikationsnetze
Dr. Erik Wilde, Institut für Technische Informatik und Kommunikationsnetze, ETH-Zentrum, CH - 8092 Zürich
fon +41-1-6325132; mobile/sms +41-79-6431851; fax +41-1-6321035; mailto:net.dret@dret.net; http://dret.net/

Aufgabenstellung Diplomarbeit SS 2001 für Alexander Karg

Im XLinkbase-Projekt entstehen ein Modell und eine Implementierung einer Datenbank, die spezifisch dafür ausgelegt ist, beliebige Ressourcen-Informationen zu sammeln (meist Referenzen auf Internet-basierte Ressourcen in Form von HTTP- oder FTP-URIs) und in komplexe inhaltliche Zusammen-hänge zu bringen. Das zugrundeliegende Modell verwendet an vielen Stellen XML-basierte Technolo-gien, um die verschiedenen Komponenten des Gesamtsystems miteinander zu verbinden. Die Grundarchitektur des Systems ist ein Client/Server-Modell, in dem der Client entweder ein normaler HTML-basierter Web-Browser, ein XML/XLink-Browser der nächsten Generation, oder auch ein spe-zialisierter
Client sein kann, der spezifisch für den XLinkbase-Server programmiert ist und die Informa-tionen in der XLinkbase in anschaulicher Weise graphisch modelliert. Der XLinkbase Server ist ein durchgängig XML-basiertes System, das über HTTP mit dem Client kommuniziert und die Requests und Responses über eine flexibel konfigurierbare Menge von XSLT Style Sheets verarbeitet. Die eigentliche Speicherkomponente ist ebenfalls austauschbar, im Normalfall wird dies jedoch entweder ein relationales Datenbanksystem oder eine XML-Lösung sein.
Basierend auf der Idee und Architektur des XLinkbase Systems (insbesondere des Datenmodells und der verwendeten XSLT Transformationen) soll in der Diplomarbeit ein eigenständiges Programm konzipiert und implementiert werden, das die Idee einer Linkdatenbank als Speicher für Metadaten auf die Archivierung von Bildern anwendet. Dieses Programm soll als stand-alone Lösung benutzbar sein, also nicht auf den XLinkbase Server angewiesen sein (wobei eine spätere Anbindung an XLinkbase-basierte Anwendungen zumindest konzeptionell berücksichtigt werden muss). Das Programm soll ins-besondere die folgenden Anforderungen erfüllen:

- Die Funktionalität muss sich an bestehenden Vorbildern zumindest orientieren, aber nicht auf die von diesen angebotenen Funktionen beschränkt sein. Insbesondere der Aspekt der saub-eren Trennung von Daten und Metadaten ist in dieser Arbeit sehr wichtig.
- Die Daten sollen gemäss einem eigenständig zu erstellenden Datenmodell (aber eng orientiert am XLinkbase-Datenmodell) strukturiert werden. Als Export- und Import-Format werden XML verwendet. Eine saubere Dokumentation des Datenmodells (idealerweise als XML Schema und nicht als DTD) ist notwendig. Flexibilität, Erweiterbarkeit und einfache Handhabbarkeit sind beim Datenmodell sorgfältig gegeneinander abzuwägen.
- Bei der Implementierung haben Robustheit, Erweiterbarkeit und ein einfaches, aber intuitives Interface Vorrang vor anderen Aspekten.

Ausgehend von diesen Punkten ist zunächst ein Grobkonzept der Architektur zu erstellen, bevor in Abstimmung mit dem Betreuer ein detailliertes Implementierungskonzept erarbeitet wird. Angesichts der Vielfalt der möglichen Varianten empfiehlt sich ein stufenweises Vorgehen mit mehreren gestaffelten Implementierungs-, Test- und Dokumentationsphasen.