Wolfgang Meier

User Needs and Digital Libraries Design (3):

A Practical Introduction to Selected Standards and Technologies

Objectives

- Present a broader view on some of the standards and technologies we use in our projects
- Provide a practical introduction to the development of XML-centered DL applications that utilize existing standards
- Design of Digital Libraries with focus on open standards, interoperability and extensibility

Outline

1. Introduction
2. Metadata Standards Overview
3. XML and Databases
4. Querying XML
5. Use Case: SozioNet

Part 1

Introduction:
Why Open Standards are Important

The user's workplace (1)

Some technologies, a (not really :-) typical sociologist may want to use:

- MS Word/Open Office for writing
- Endnote to maintain references
- Library catalog to check relevant literature and citations
- Dictionaries, thesauri, classifications

The user's workplace (2)

Personal information management:

- A Wiki to collect/organize notes and text snippets
- A Blog to post news and thoughts
- Newsletter and mailing list subscriptions
- Search engines / databases

The user's workplace (3)

Research tools:

- Software for statistical analysis of empirical research data (SPSS)
- Content analysis system

The user's workplace (4)

- Can all these technologies be bound together?
- There's no single application to meet all user demands:
  - Needs are highly context dependent and never fixed
  - Vary extremely between different scientific domains
  - Every discipline constitutes its own “community of practice”, with its own rules, style and methodology

The user's workplace (5)

- But: most of these applications already support interoperability through standard protocols, e.g.:
  - Document authoring: WebDAV
  - Bibliographic software: Z39.50
  - Library catalog: Z39.50
  - Blog/Wiki: RSS, Atom

Open Standards for Document Authoring

- PDF (?)
- LaTeX, HTML
- Docbook, TEI, Open Office XML
- SMIL (multimedia presentations)
- SVG (vector graphics), MathML
- And many more ...

Areas of Standardization

1) Metadata

2) Protocols

3) Document and data formats for storage, archiving and exchange

Part 2

Metadata

What is Metadata?

- Metadata standards form a common, shared vocabulary to describe information objects
- Described aspects may be intrinsic or extrinsic to a resource:
  - Content: what the object contains
  - Context: who created the object, when, how
  - Structure: how is it related to other objects

Some Metadata Standards

- Established library standards: MARC (USMARC, UniMARC), MAB ...
- Dublin Core
- RDF + RDF Schema + OWL (define a basic model + syntax, not a vocabulary)
- MODS + METS

Resource Description Framework (RDF)

- General-purpose language
- Not limited to bibliographic metadata
- Provides a basic model and serialization syntax, not a vocabulary
- Vocabulary semantics have to be defined by the community in terms of RDF Schema

RDF Schema (RDFS)

- Vocabulary description language
- Defines modeling primitives for
  - classes
  - properties
  - relationships between classes and properties
  - restrictions
- Ensures the validity and data integrity of a given data set

RDF: Some Benefits

- Metadata designers can rely on a common foundation of established schemas (DC, vCard …)
- Domain specific aspects can be made explicit through RDF Schema
- Vocabulary description can be accessed by users and software agents
- Reference resources anywhere on the web

Example: RDF Metadata (SozioNet)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:sn="http://www.sozionet.org/1.0/#" xmlns:dcq="http://purl.org/dc/terms/#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#
    <sn:ResearchPaper
        rdf:about="http://delos-noe.iei.pi.cnr.it/activities/internationalforum/Actors-Roles.pdf">
        <dc:title>Reference Models for Digital Libraries: Actors and Roles</dc:title>
        <dcq:alternative>DELOS/NSF Working Group: Final Report</dcq:alternative>
        <dcq:abstract>This report summarizes the discussion of the DELOS/NSF Working Group on
        "Reference Models for Digital Libraries: Actors and Roles".</dcq:abstract>
        <dc:language>EN</dc:language>
        <dcq:IMT>application/pdf</dcq:IMT>
        <dcq:created>2003-07-23</dcq:created>
        <dc:creator>Borbinha, José</dc:creator>
        <dc:creator>Kunze, John</dc:creator>
        <dc:creator>Lieder, Hans-Jörg</dc:creator>
        <dc:creator>Mabe, Michel</dc:creator>
        <dc:creator>Mutschke, Peter</dc:creator>
        <dc:creator>Besser, Howard</dc:creator>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/classification#50200"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/classification#29900"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Analyse"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Informatik"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Informationswissenschaft"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Modellentwicklung"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#soziales_Netzwerk"/>
    </sn:ResearchPaper>
</rdf:RDF>
```

RDF: Disadvantages

- Looks like XML, but is not XML
- RDF/XML is just one serialization form of RDF
- One RDF data set maps to multiple XML serializations
- XML and RDF are different worlds!

RDF: Disadvantages

- Need proper tools to process RDF
- Examples: Jena, Redland ...
- Negative for XML-centered applications:
- Different technologies required for processing metadata (RDF) and data (XML documents)

MODS

- Created and maintained by the LoC
- Bibliographic element set oriented towards library applications
- Expressed in XML Schema
- Elements map to MARC fields
- Semantics compatible with MARC

MODS: Advantages

- Defines a bibliographic vocabulary that is familiar to librarians
- Based on established library standards
- Allows some degree of freedom in specifying fields
- XML Schema based: can be processed and queried with XML tools

MODS: Example

```
<?xml version="1.0" encoding="utf-8"?>
  <mods xmlns="http://www.loc.gov/mods/v3" ID="one">
    <name type="personal">
      <namePart type="given">Don</namePart>
      <namePart type="family">Mitchell</namePart>
      <role><roleTerm type="text">author</roleTerm></role>
    </name>
    <titleInfo>
      <nonSort>The</nonSort>
      <title>Right to the City</title>
      <subTitle>Social Justice and the Fight for Public Space</subTitle>
    </titleInfo>
    <originInfo>
      <dateIssued>2003</dateIssued>
      <issuance>monographic</issuance>
    </originInfo>
    <typeOfResource>text</typeOfResource>
    <genre>book</genre>
    <subject>
      <topic>public space</topic>
      <topic>power</topic>
      <geographic>United States</geographic>
    </subject>
    <recordInfo>
      <recordCreationDate encoding="w3cdtf">2003-12-11</recordCreationDate>
      <recordIdentifier source="citekey">MitchellD2003a</recordIdentifier>
    </recordInfo>
  </mods>
```

MODS: Advantages

- Naturally uses other XML standards like XLink, e.g. to link to authority files
- Extensible: elements from other schemas can be embedded
- Usable within METS records (administrative and structural metadata)

MODS: Disadvantages

- Not a general-purpose format: targeted at bibliographic metadata
- Allows alternative ways to specify elements: sometimes difficult to process
- Relatively new standard: usage possibilities still need to be explored

Beyond Bibliographic Metadata

- Example: MPEG-7
- Content based description standard for various types of audiovisual information
- Describes things like:
  - Life-cycle of multimedia content
  - Storage format, media quality, location
  - Creation and production process of the source
  - Spatial, temporal and media source structure

Part 2

XML and Databases

Why XML Rules (1)

- Contrary to HTML or LaTeX, XML defines a meta-syntax upon which domain-specific markup languages are build
- Separation of structure, content and layout
- Integration of text and data, highly structured and semi-structured contents
- A wealth of related standards: XSLT, XQuery, XLink, XInclude

Why XML Rules (2)

- Heterogeneity: information does not need to be forced into tables and columns
- Extensibility: XML embraces change instead of avoiding it
- Flexibility: data may vary in size and structure from instance to instance

XML Storage Options

- Simple file or CLOB storage
- Map XML data to a relational model
- Use an XML-enabled RDBMS
- Native XML Database (NXDB)

Products

- XML-enabled databases: DB2, Oracle
- Middleware: XQuark Fusion, XML-DBMS ...
- Native XML Databases:
  - Commercial: Tamino, XStreamDB, MarkCIS ...
  - Open Source: XIndice, Berkeley XML, dbXML, eXist

Native XML DB: Advantages

- Offer all the flexibility and extensibility of XML
- Store and query arbitrary XML data structures
- In most products, not even a schema is required
- Big advantage if data structures are heterogenous and change often

Native XML DB: Advantages

- Tightly integrated with related XML standards and XML development tools
- Rapid application development: entire applications can be developed just using XML and related standards

BUT:

- Greater flexibility has to be paid by performance restrictions, size limitations etc.
- Native XML DB should NOT be seen as a replacement for a relational database
- Just converting tables to XML is a bad idea
- Designing a good schema for XML documents needs skills and experience

Introducing eXist

http://exist-db.org

- Started in 2001
- Inspired by:

  D. Shin, H. Jang, H. Jin: „*BUS: An Effective Index and Retrieval Scheme in Structured Documents*“. In Proceedings of the 3rd ACM International Conference on Digital Libraries, 1998, Pittsburgh, PA.

Features

- Native XML Data Store
- Model-Oriented: XML nodes are stored in a persistent DOM
- Resources (XML or binary) are managed in hierarchical collections, much like in a file system
- Unix-like access rights, authentication

Features

- Automatic indexing of stored resources
  - Structural index: indexes elements and attributes; always created automatically
  - Fulltext index: indexes text tokens; can be switched off for defined document parts
- XUpdate: partial document updates
- Interfaces: XML-RPC, SOAP, REST-style, WebDAV, XML:DB API

Deployment

1) Stand-alone server
2) Embedded library
3) Servlet environment, integrated with Cocoon / Jboss

- Runs from read-only media
- In Java, simple switch from embedded to remote servers

XQuery

- Implementation based on Nov. 2003 working draft
- Passes 92% of the official use-cases
- XML Schema related features missing
- Large set of additional modules for fulltext search, web integration, database management

Web Application Development

- Write entire web applications just in XQuery (example: SozioNet)
- Integration with Cocoon offers unlimited possibilities for modularized, XML-based applications

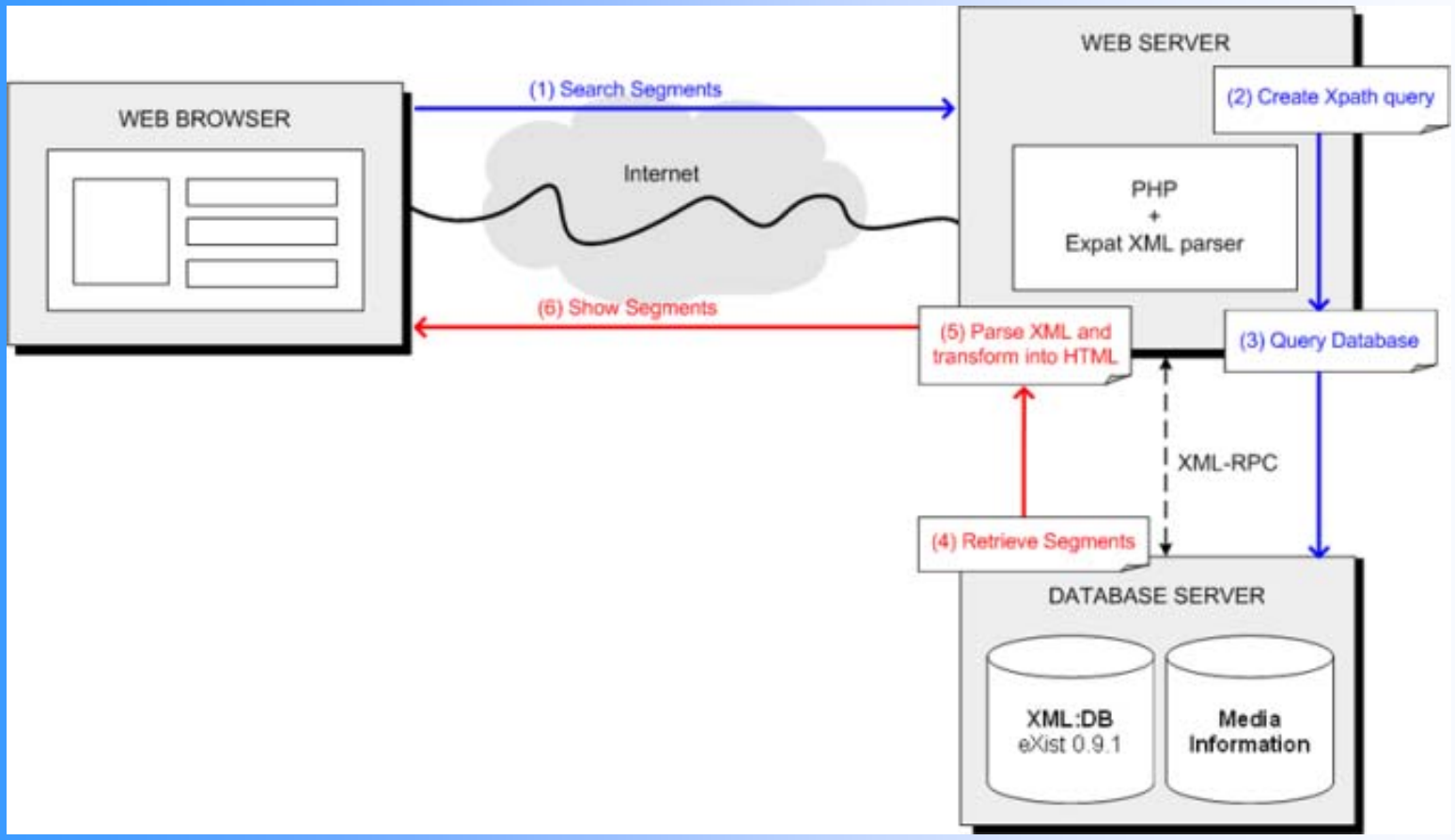
Application Examples

- Two examples taken from a recent survey in the eXist user community
- Both combine multimedia resources, MPEG-7 metadata and eXist XML database:
  - OpenDrama
  - Xunami™

OpenDrama

- “The Digital Heritage of Opera in the Open Network Environment”
- Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
- “Novel platform to author and deliver rich cross-media digital objects of lyric opera and other vocal dramatic music”

MPEG-7 Web Browser

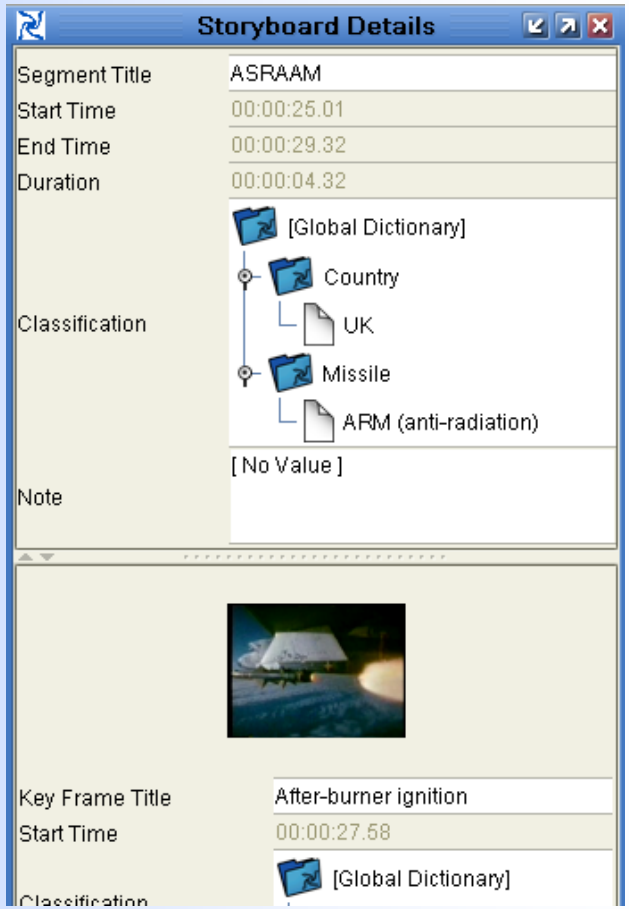


Graphics by Oscar Celma, used with permission

Xunami™

- Commercial application developed by ***Divstrat Pty Ltd, Sydney, Australia***
- Multi-lingual, web-based, client-server application
- ***Indexing, querying and retrieval of digital multimedia in distributed repositories***
- ***Supports still and moving images***

Xunami™



Graphics by Divstrat Pty Ltd., reproduced with permission

Part 3

Querying XML

XQuery

- Often compared to SQL
- But XQuery is more than a query language, it is a complete, functional programming language
- Designed from the start to mix XML fragments with a functional language
- XQuery is a typed language (based on XML Schema)

XQuery Basics

The XQuery prolog required for the following examples:

```
xquery version “1.0”;

declare namespace mods=”                                    ";
```

Basic XPath

Simple path selection:

```
/mods:mods/mods:titleInfo/mods:title
```

Descendant selection:

```
/mods:mods//mods:title
```

Using a wildcard:

```
/mods:mods/mods:*/mods:title
```

Changing the context

Change the context document set:

```
collection("/db/meta")//mods:mods
```

```
doc("/db/meta/12345.mods")//mods:mods
```

Filter Predicates

Select the first name element:

```
//mods:mods/mods:name[1]
```

Select record with title containing “justice”:

```
//mods:mods[contains(mods:titleInfo/mods:title, “city”)]
```

Select record with title starting with “social”:

```
//mods:mods[starts-with(mods:titleInfo/mods:title, “social”)]
```

Boolean Operators

Select record written by “Mitchell”:

```
//mods:mods[mods:name[mods:namePart = “Mitchell”] and
            contains(mods:titleInfo/mods:title, “city”)]
```

Select record written by “Mitchell” or “Castells”:

```
//mods:mods[mods:name[mods:namePart = “Mitchell” or
            mods:namePart = “Castells”]]
```

Constructing XML Output

List all titles by Castells:

```
<titlesByAuthor>
{
    //mods:mods[
        mods:name[mods:namePart = “Castells”]
    ]/mods:titleInfo
}
</titlesByAuthor>
```

FLWOR Expressions (1)

Use “for” to iterate over all items in the specified sequence:

```
for $record in //mods:mods[mods:name/mods:namePart = “Castells”]
return <result>{$record}</result>
```

“let” assigns the results of an expression to a variable (without starting an iteration):

```
let $hits := //mods:mods[mods:name/mods:namePart = “Castells”]
return
    <result>{$hits}</result>
```

Sorting

Sort the records by title:

```
for $record in //mods:mods[mods:name/mods:namePart = “Castells”]
order by $record/mods:titleInfo/mods:title
return <result>{$record}</result>
```

Multiple sort criteria can be specified:

```
for $record in //mods:mods[mods:name/mods:namePart = “Castells”]
order by $record/mods:titleInfo/mods:title,
  $record/mods:titleInfo/mods:subTitle
return <result>{$record}</result>
```

FLWOR Expressions (2)

Combining “let” and “for”:

```
let $hits := //mods:mods[mods:name/mods:namePart = “Castells”]
return
  <results hits=”{count($hits)}”>
  {
    for $record in $hits
    let $uri := document-uri($record)
    order by $record/mods:titleInfo/mods:title
    return
      <hit uri=”{$uri}”>
        <title>{$record/mods:titleInfo/mods:title}</title>
        <year>{$record/mods:originInfo/mods:dateIssued}</year>
      </hit>
  }
  </results>
```

Functions

```
declare function local:display($record as element()) as element()
{
  <hit uri="{document-uri($record)}">
    <title>{$record/mods:titleInfo/mods:title}</title>
    <year>{$record/mods:originInfo/mods:dateIssued}</year>
  </hit>
};

let $hits := //mods:mods[mods:name/mods:namePart = "Castells"]
return
<results hits="{count($hits)}">
  {
    for $record in $hits
    order by $record/mods:titleInfo/mods:title
    return local:display($record)
  }
</results>
```

Part 4

Use Case: SozioNet

SozioNet

- Launched in 04/2002
- Member of http://www.infoconnex.de:
- A joint effort to integrate existing services and ongoing activities related to education, social science and psychology
- Funded by the German Federal Ministry of Education and Research

Goals

- Inspired by MathNet, PhysNet, SOSIG ...
- Based on the principle of self-organization of social science institutions and scientists
- Access to freely available web resources with relevance to German social science
- Provide domain specific services for a given scientific community, while still being committed to open standards

Goals for the Community

- Create a network of social science institutions and scientists
- Establish common standards for the publication of social science resources on the web:
  - Formal requirements
  - Metadata sets and quality of metadata
  - Use of established classifications and thesauri

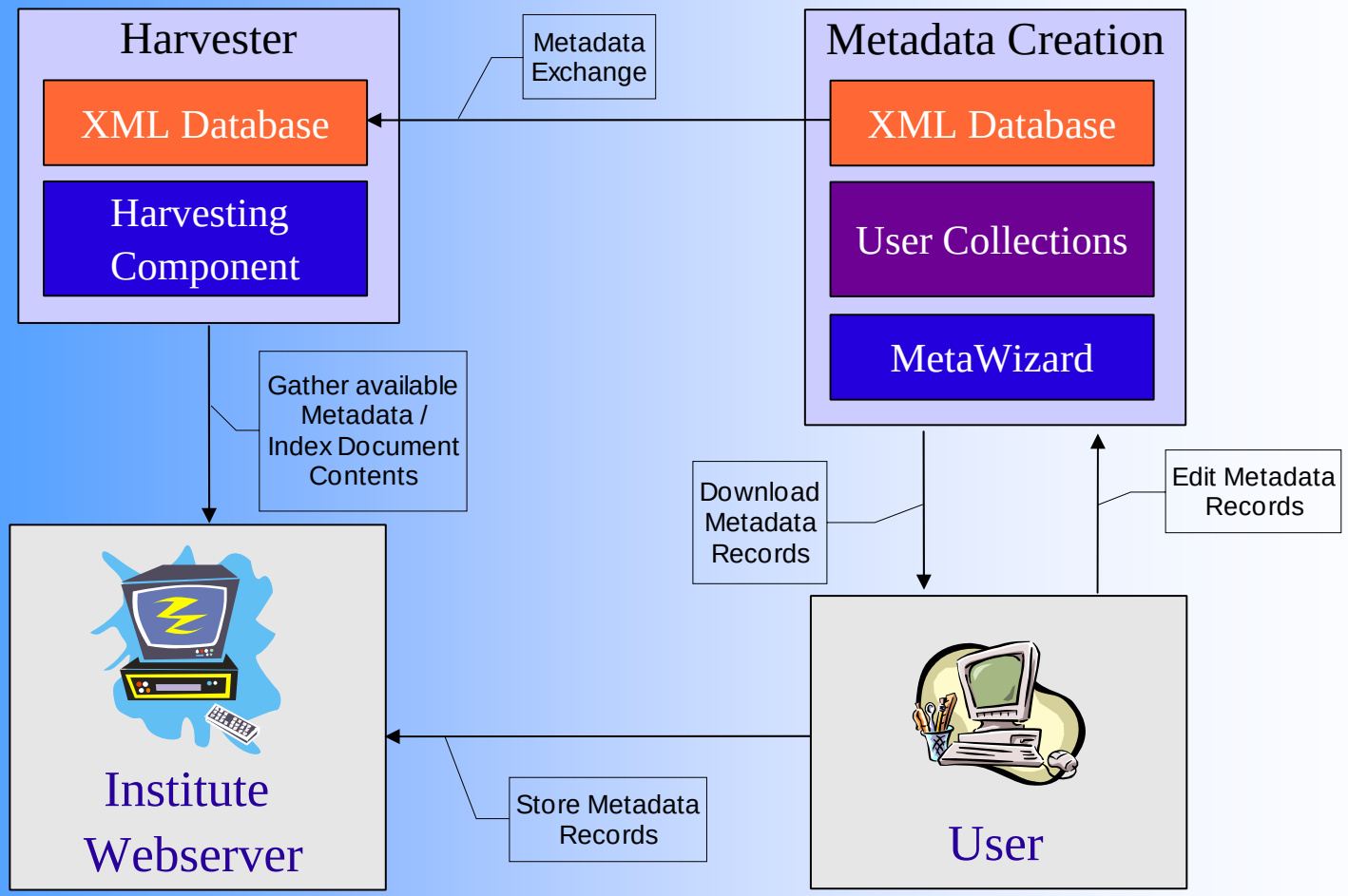
Technical Goals

- SozioNet helps authors create semantically rich metadata
- Implements a general infrastructure to harvest relevant resources from the web
- Provides domain specific search services
- Integrates with and complements other services through Infoconnex: scientific databases, printed resources, library services

Software Tools

- MetaWizard for metadata creation:
  - Personalized user interface
  - Integrated thesaurus and classification browser
  - Based on Cocoon (XML Application Framework)
- Harvester:
  - Collect metadata and fulltext from the web
  - Entirely based on XML and related standards

Metadata Workflow



Metadata (1)

- Based on RDF + RDFS
- Most elements in the metadata model are covered by common vocabularies like Dublin Core
- But some aspects are specific to the social science domain
- Domain specific aspects should be made transparent to end-users and software agents

Metadata (2)

- Metadata records in SozioNet are backed by an OWL Ontology
- Contains all concepts not covered by Dublin Core or other schemes
- Defines a shared vocabulary for the SozioNet domain

Metadata (3)

- Main areas defined by the ontology:
  - Class hierarchy for different types of resources and their properties
  - Thesaurus and classification terms
  - Organizations, institutional and personal information

Metadata Example

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:sn="http://www.sozionet.org/1.0/#" xmlns:dcq="http://purl.org/dc/terms/#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#
    <sn:ResearchPaper
        rdf:about="http://delos-noe.iei.pi.cnr.it/activities/internationalforum/Actors-Roles.pdf">
        <dc:title>Reference Models for Digital Libraries: Actors and Roles</dc:title>
        <dcq:alternative>DELOS/NSF Working Group: Final Report</dcq:alternative>
        <dcq:abstract>This report summarizes the discussion of the DELOS/NSF Working Group on
        "Reference Models for Digital Libraries: Actors and Roles".</dcq:abstract>
        <dc:language>EN</dc:language>
        <dcq:IMT>application/pdf</dcq:IMT>
        <dcq:created>2003-07-23</dcq:created>
        <dc:creator>Borbinha, José</dc:creator>
        <dc:creator>Kunze, John</dc:creator>
        <dc:creator>Lieder, Hans-Jörg</dc:creator>
        <dc:creator>Mabe, Michel</dc:creator>
        <dc:creator>Mutschke, Peter</dc:creator>
        <dc:creator>Besser, Howard</dc:creator>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/classification#50200"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/classification#29900"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Analyse"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Informatik"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Informationswissenschaft"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#Modellentwicklung"/>
        <dc:subject rdf:resource="http://www.sozionet.org/1.0/thesaurus#soziales_Netzwerk"/>
    </sn:ResearchPaper>
</rdf:RDF>
```

MetaWizard: Features

- Step-by-step wizard spanning multiple screens
- Summarizing feature: new resources are scanned for existing metadata (e.g. DC META tags in HTML), which is then reviewed by the user
- Based on Cocoon2 and XMLForms: Model-View-Controller architecture
- Personalized User Interface

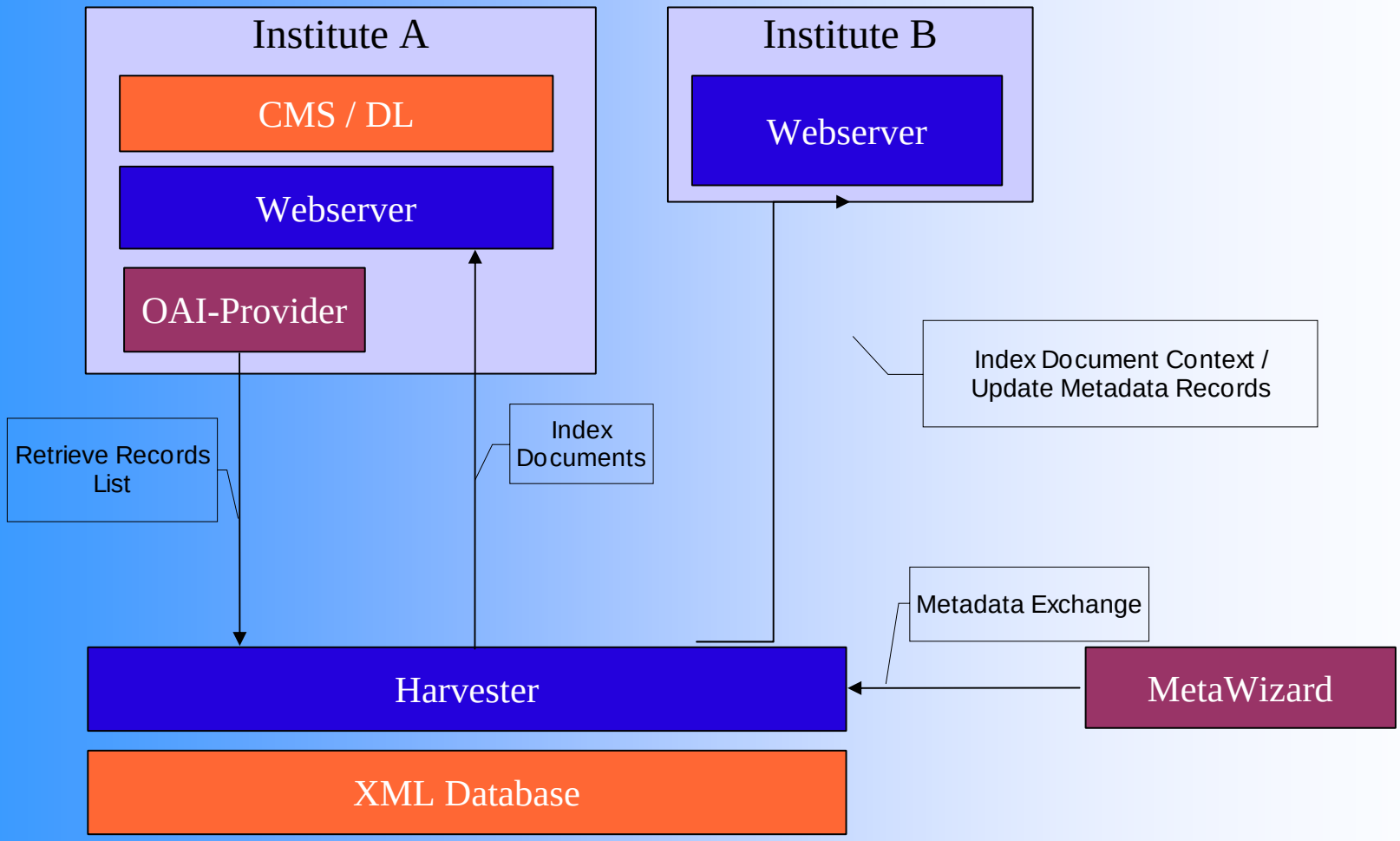
MetaWizard: Features

- Integrated thesaurus and classification browser
- Backed by a native XML database
- Distributable as one, single package (.war file)

Personalization

- Each user has his own home-collection on the server
- Metadata records persist on the database
- Records entered through previous sessions can be revised or removed at any time
- User authentication is done by the XML database

Metadata Harvesting



Metadata Harvesting

- Harvesting process uses information entered through the MetaWizard as a starting point by scanning each URL referenced in the record
- Local metadata found on web server is added
- Project partners can report additional URLs to be scanned

Harvesting: Requirements

- RDF data should not be touched/altered by the harvester
- Metadata structure may vary in domain specific aspects (e.g. additional documentation languages)
- XML documents should be preserved
- But: harvester also has to deal with non-wellformed HTML, PDF, PS ...

SozioNet Harvester (1)

- The core paradigm of Cocoon2 applied to a harvesting scenario!
- Entirely based on XML and related standards
- Uses freely available tools
- Backed by a simple, yet powerful component model

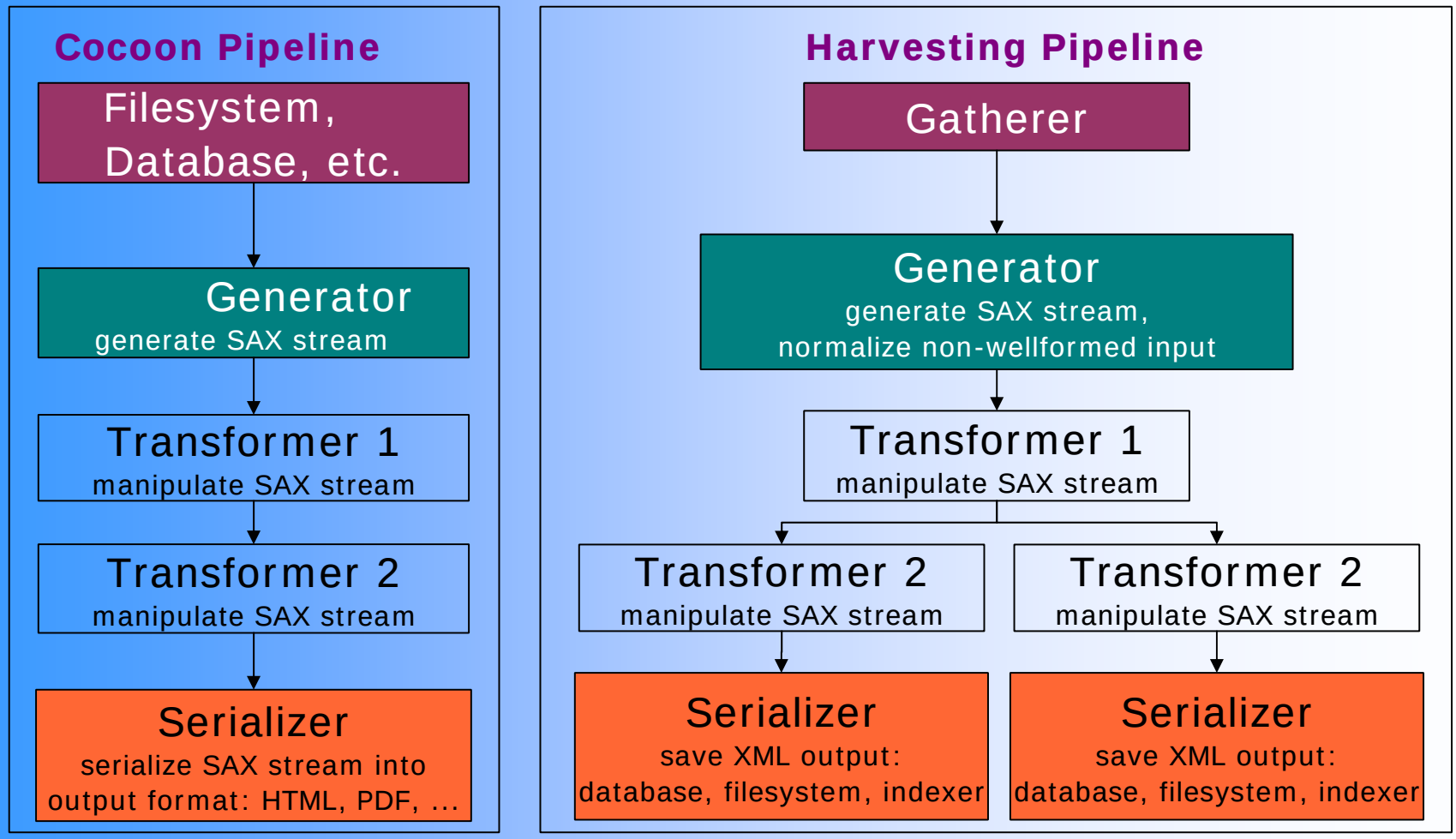
SozioNet Harvester (2)

- The core paradigm of Cocoon2: the pipeline
- Basic components:
- Generator -> $Transformer_1$...$Transformer_n$-> Serializer
- Components operate on SAX streams

SozioNet Harvester (3)

- The driving force behind Cocoon: Avalon component framework based on roles and contracts
- Harvester uses the same framework!
- All input is transformed into XML
- Highly configurable: new components may be added at any time

Pipelines Compared



Searching

- Search facilities implemented on top of XML database
- Not an ideal solution: database just deals with the serialization form of RDF
- But:
  - Using an XML DB is fast and simple
  - Structured content is preserved
  - Advanced RDF features are not used