OAI-PMH Implementation

- Tutorial -

Uwe Müller
Humboldt University Berlin

In the Beginning: Thanks!

- Some of the slides presented here are my own!
- Many of them have been kindly donated by (taken from!):
  - Andy Powell
  - Herbert Van de Sompel
  - Carl Lagoze
  - Hussein Suleman
  - Michael Nelson
  - Simeon Warner
  - Heinrich Stamerjohanns
  - Pete Cliff
  - (and others probably...)

Coverage

- Introduction to the main ideas of the OAI-PMH
- A detailed view into the protocol specification
- Example Implementation of an OAI Data Provider
- Considerations for the development of OAI Service Providers
- Metadata description in XML: What if I need more than Dublin Core?

What you will learn during next 3 hrs.

- The functioning of the OAI-PMH in detail
- The principle functioning of OAI Data and Service Providers
- The requirements and necessary considerations for implementing OAI Data and Service Providers
- The principle approach for implementing a Data Provider - from scratch - using existing tools
- How to proceed when deploying another metadata format to be used with OAI

Agenda

Part I - History and Overview

Part II - OAI Serviceprovider - Examples

Part III - Technical Introduction

Part IV - Implementation Issues

Part V - Different Metadata Formats

Tutorial
Open Archive Initiative

Part I
History and Overview

OAI Roots…

- the roots of OAI lie in the development of eprint archives…

  arXiv, CogPrints, NACA (NASA), RePEc, NDLTD, NCSTRL

- each offered Web interface for deposit of articles and for end-user searches
- difficult for end-users to work across archives without having to learn multiple different interfaces
- recognised need for single search interface to all archives

  Universal Pre-print Service (UPS)

Searching vs. Harvesting

- two possible approaches to building the UPS…
  1. cross searching multiple archives based on protocol like Z39.50
  2. harvesting metadata into one or more ‘central’ services – bulk move data to the user-interface
- US digital library experience in this area (e.g. NCSTRL) indicated that cross searching not preferred approach - distributed searching of N nodes viable, but only for small values of N
- NCSTRL: N > 100; bad

Problems of Cross Searching

- collection description
  - How do you know which targets to search?
- query-language problem
  - Syntax varies and drifts over time between the various nodes.
- rank-merging problem
  - How do you meaningfully merge multiple result sets?
- performance
  - tends to be limited by slowest target
  - difficult to build browse interface

Universal Preprint Service

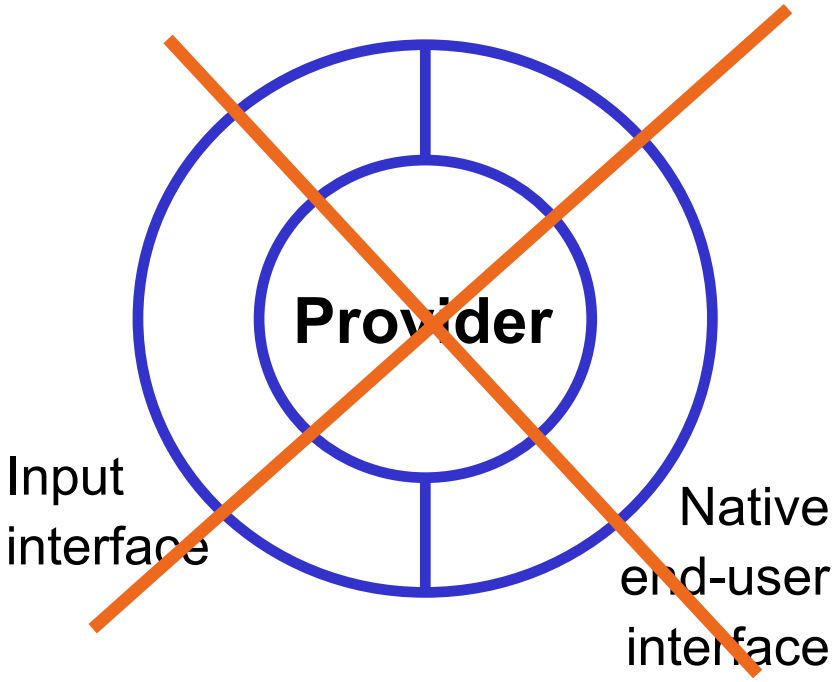
- a cross-archive Digital Library that provides services on a collection of metadata harvested from multiple archives
  - based on NCSTRL+; a modified version of Dienst
- demonstrated at Santa Fe NM, October 21-22, 1999
  - http://ups.cs.odu.edu/
  - D-Lib Magazine, 6(2) 2000 (2 articles)
    http://www.dlib.org/dlib/february00/02contents.html
- UPS was soon renamed the Open Archives Initiative (OAI) http://www.openarchives.org/

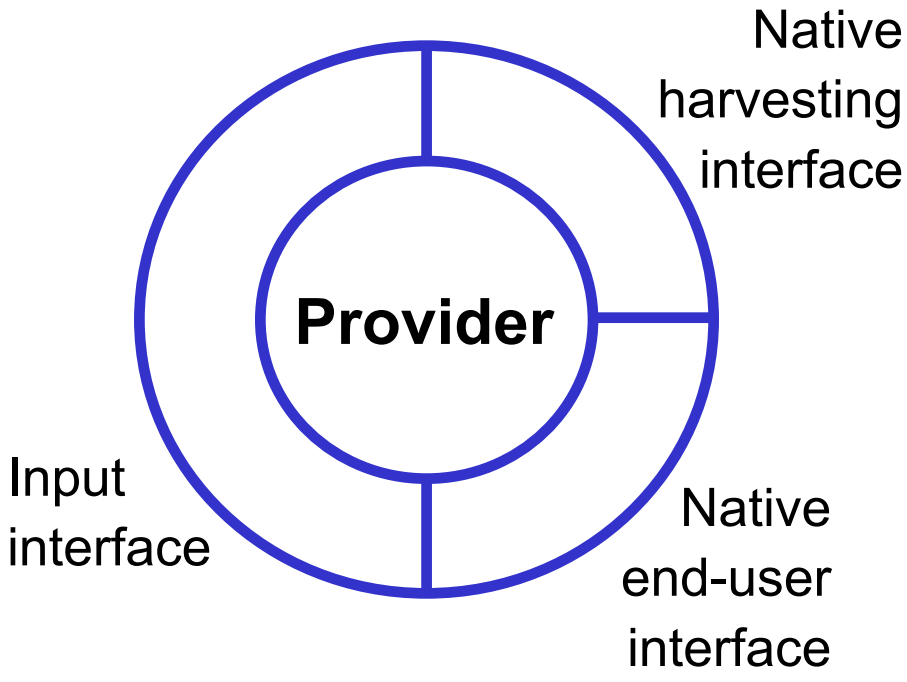
Data and Service Providers

- UPS identified two logical groups of services…
- data providers
  - handle deposit/publishing of resources in archive
  - expose metadata about resources in archive
- service providers
  - harvest metadata from data providers
  - use it to offer single user-interface across all harvested metadata
- note:
  - data provider may also be responsible for human-oriented (i.e. Web) interface to archive
  - both functions may be offered by same ‘service’

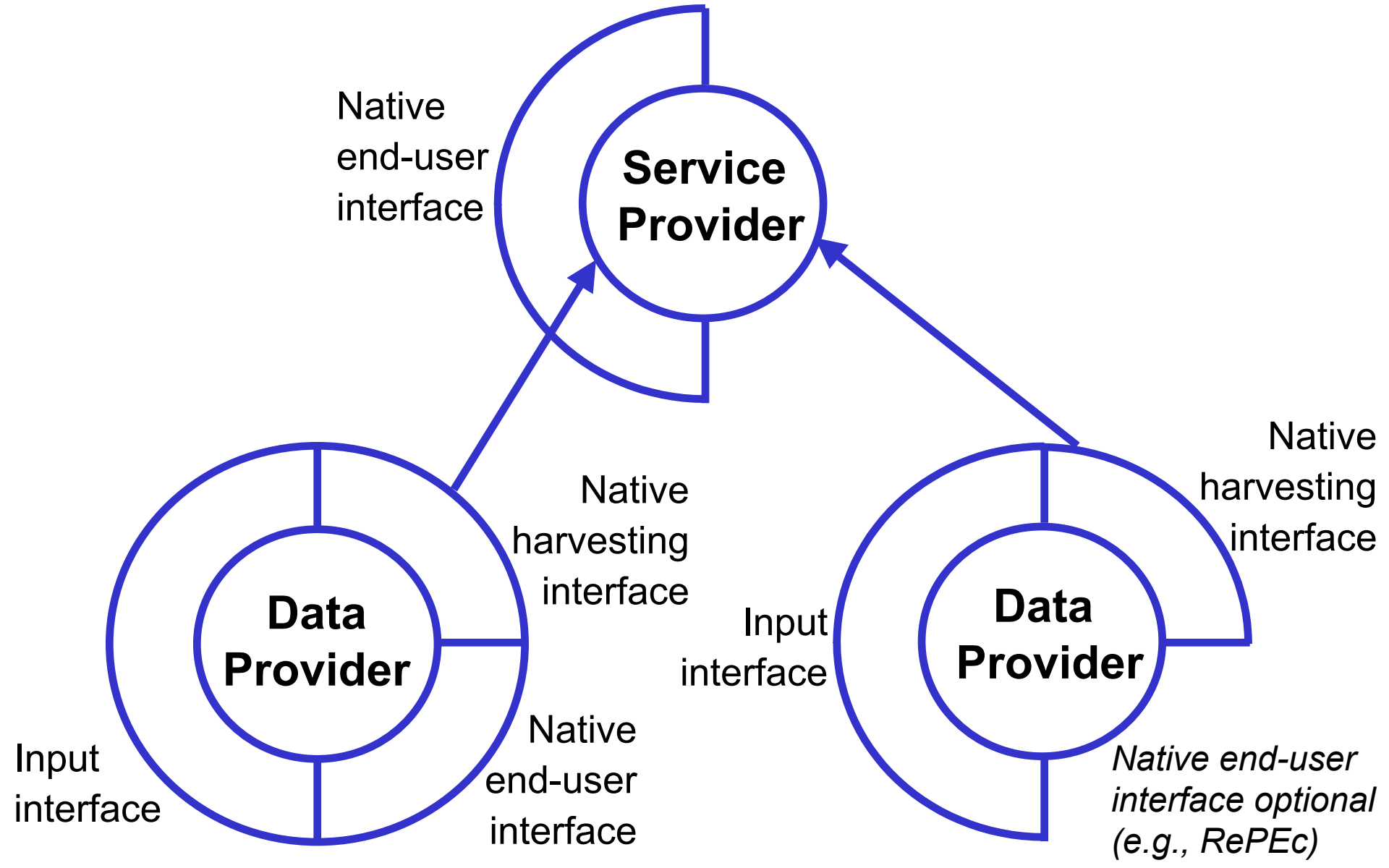
Human vs. Machine Interfaces

- move away from only supporting human end-user interfaces for each archive …
- … to supporting both, human end-user interface **and** machine interfaces for harvesting





Service Provider Harvesting



Metadata Harvesting Requirements

- in order to allow the harvesting approach to work we need agreements about …
  - transport protocols – HTTP vs. FTP vs. …
  - metadata formats – DC vs. MARC vs. …
  - quality assurance – mandatory elements, mechanisms for naming of people, subjects, etc., handling duplicated records, best-practice
  - intellectual property and usage rights – who can do what with the records
- work in this area resulted in the “Santa Fe Convention”

Santa Fe Convention [02/2000]

- goal: optimize discovery of e-prints
- inputs…
  - UPS prototype
  - RePEc/SODA “data provider / service provider” model
  - Dienst protocol
  - deliberations at Santa Fe meeting [10/1999]

OAI-PMH v 1.0 [01/2001]

- goal: optimise discovery of document-like objects
- inputs…
  - Santa Fe Convention
  - various DLF meetings on metadata harvesting
  - deliberations at Cornell
  - alpha-testers of OAI-PMH v 1.0
  - recognition of DC as ‘best’ core metadata format for interoperability across multiple archives

OAI-PMH v 1.0 [01/2001]

- low-barrier interoperability specification
- metadata harvesting model: data provider / service provider
- focus on document-like objects
- autonomous protocol
- HTTP based
- XML responses
- unqualified Dublin Core
- experimental: 12-18 months

OAI Timeline before v. 2.0

- October 21-22, 1999 - initial UPS meeting
- February 15, 2000 - Santa Fe Convention published in D-Lib Magazine

  recursor to the OAI metadata harvesting protocol
- June 3, 2000 - workshop at ACM DL 2000 (Texas)
- August 25, 2000 - OAI steering committee formed, DLF/CNI support
- September 7-8, 2000 - technical meeting at Cornell University

  defined the core of the current OAI metadata harvesting protocol
- September 21, 2000 - workshop at ECDL 2000 (Portugal)

OAI Timeline before v. 2.0

- November 1, 2000 - Alpha test group announced (~15 organizations)
- December 2000 DINI Jahrestagung in Dortmund
- January 23, 2001 - OAI protocol 1.0 announced, OAI Open Day in the U.S. (Washington DC)

  purpose: freeze protocol for 12-16 months, generate critical mass
- February 26, 2001 - OAI Open Day in Europe (Berlin)
- July 3, 2001 - OAI protocol 1.1 announced

  to reflect changes in the W3C's XML latest schema recommendation
- September 8, 2001 - workshop at ECDL 2001 (Darmstadt)

OAI-PMH v.2.0 [06/2002]

- goal: recurrent exchange of metadata about resources between systems
- inputs:
  - OAI-PMH v.1.0
  - feedback on OAI-implementers
  - deliberations by OAI-tech [09/01 - 06/02]
  - alpha test group of OAI-PMH v.2.0 [03/02 - 06/02]
  - officially released June 14, 2002

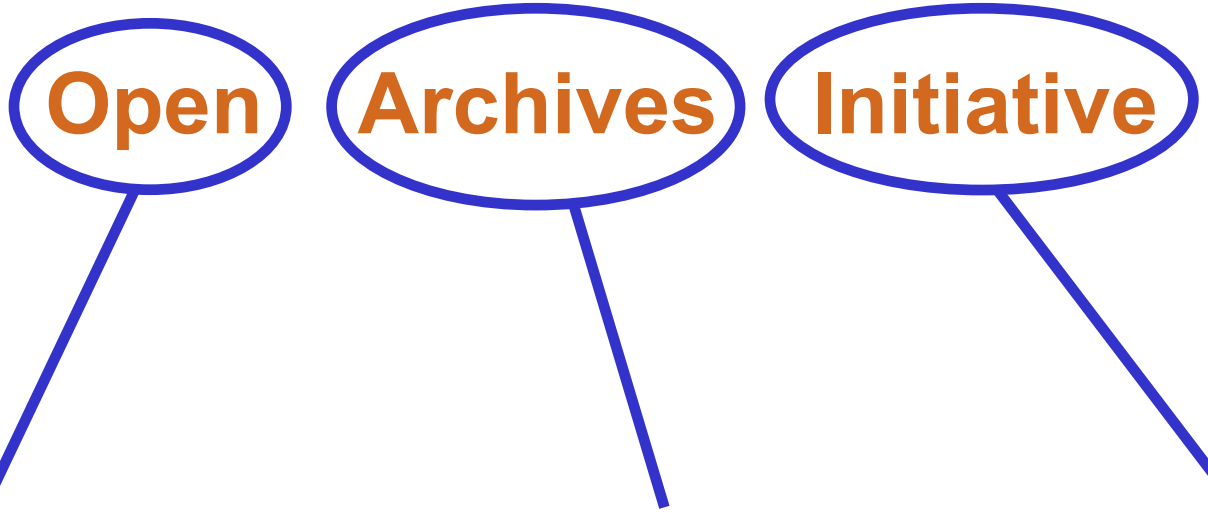
OAI-PMH v.2.0 [06/2002]

- low-barrier interoperability specification
- metadata harvesting model: data provider / service provider
- metadata about resources
- autonomous protocol
- HTTP based
- XML responses
- unqualified Dublin Core
- stable

OAI-PMH: Version Characteristics

	Santa Fe convention	OAI-PMH v.1.0/1.1	OAI-PMH v.2.0
nature	experimental	experimental	stable
verbs	Dienst	OAI-PMH	OAI-PMH
requests	HTTP GET/POST	HTTP GET/POST	HTTP GET/POST
responses	XML	XML	XML
transport	HTTP	HTTP	HTTP
metadata	OAMS	unqualified Dublin Core	unqualified Dublin Core
about	eprints	document like objects	resources
model	metadata harvesting	metadata harvesting	metadata harvesting

What's in the Name?



The protocol is openly documented, and meta-data is "exposed" to at least some peer group. (note: rights management can still apply!)

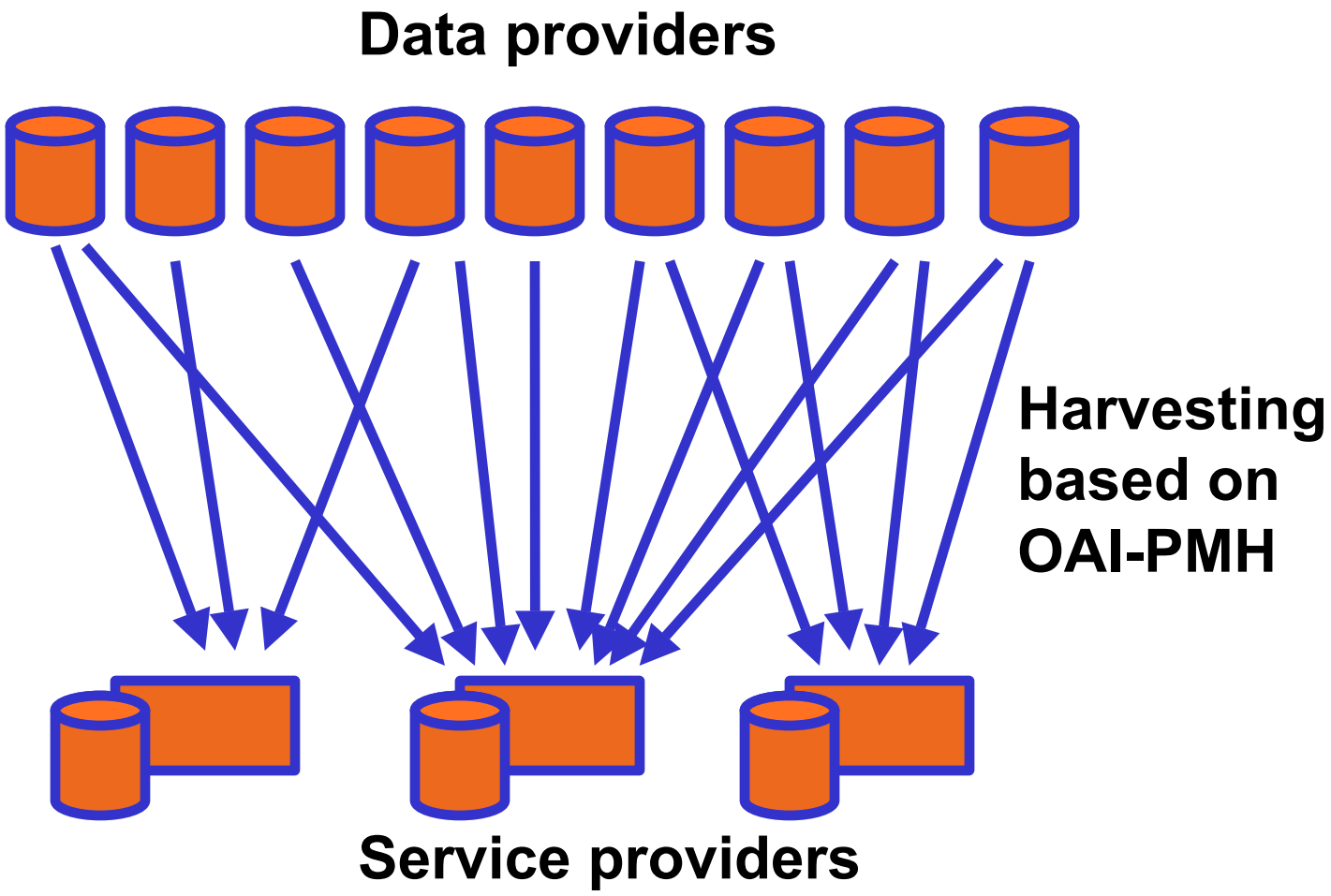
Archive defined as a "collection of stuff" -- not the archivist's definition of "archive". "Repository" used in most OAI documents.

OAI is happening at break-neck speed ...

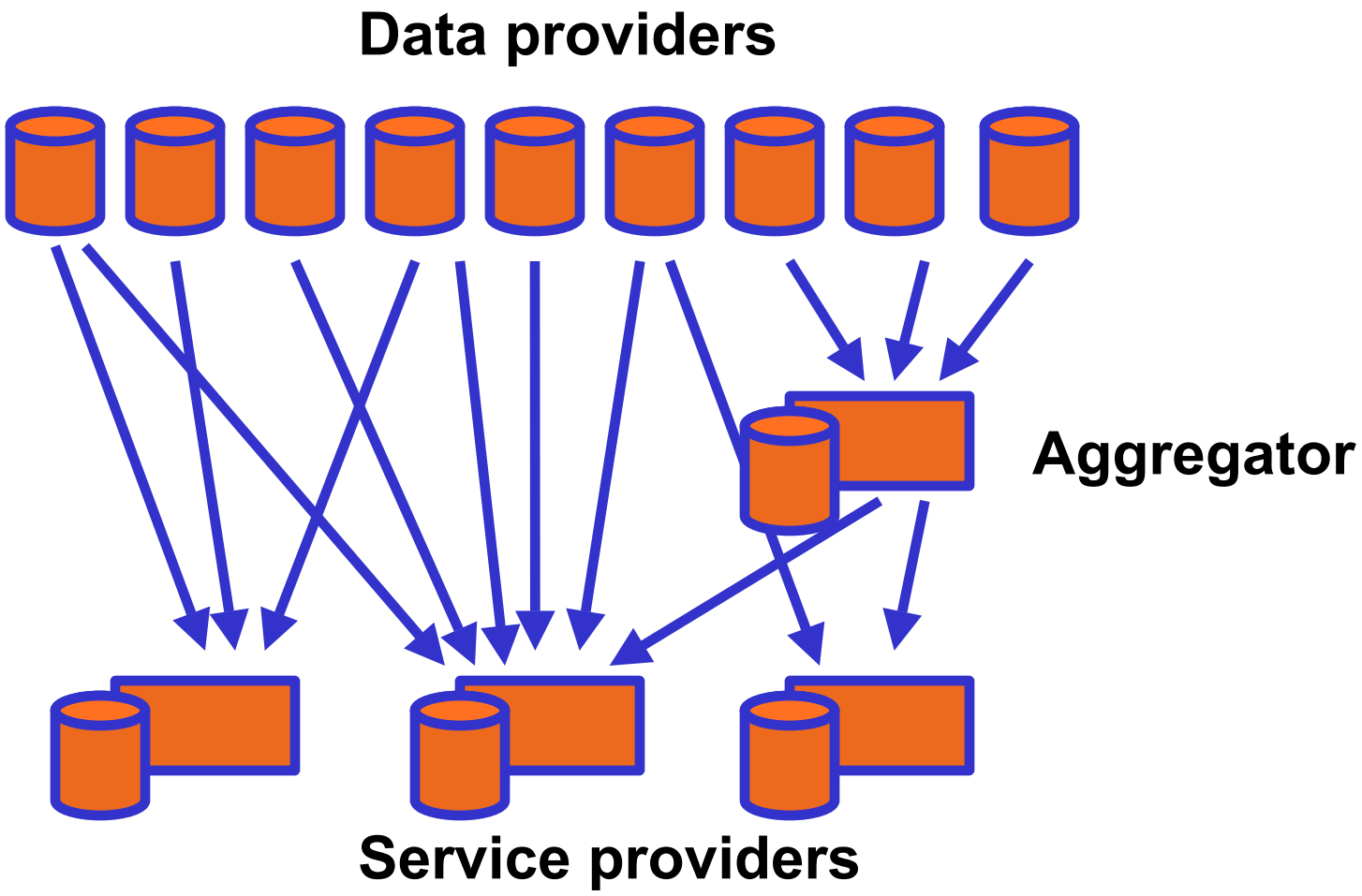
Flexible Deployment

- simple protocol based on HTTP and XML allows for rapid deployment
- a number of toolkits available
- systems can be deployed in variety of configurations
- multiple service providers can harvest from multiple data providers
- aggregators can sit between data and service providers
- harvesting approach can be complemented with searching based on Z39.50 or similar protocols

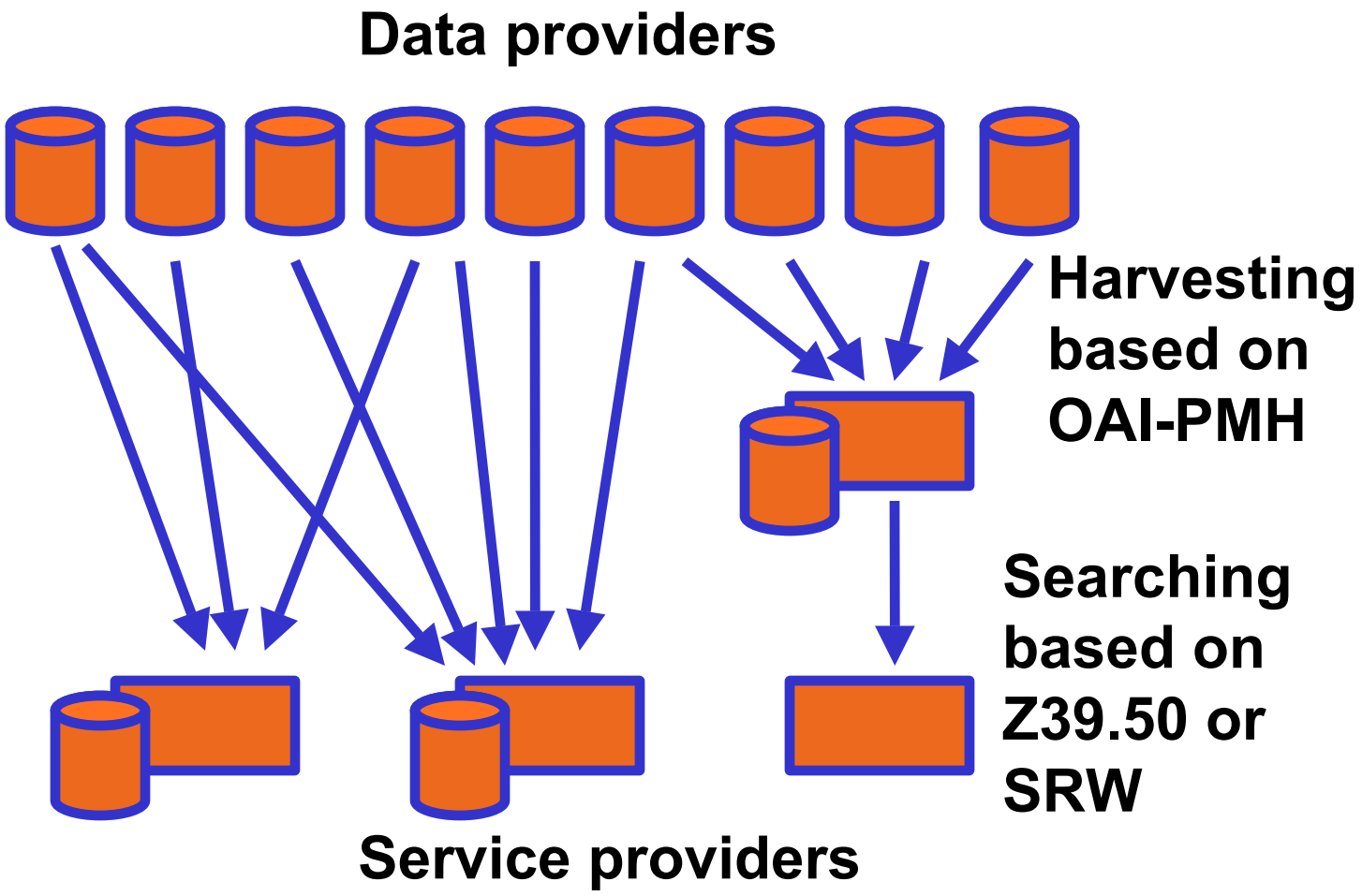
Multiple Data and Service P's



Aggregators



Can be mixed with x-Searching



Summary

- OAI-PMH – OAI Protocol for Metadata Harvesting
- low-cost mechanism for harvesting metadata records from one system to another
  - from ‘data providers’ to ‘service providers’
- development over last 2-3 years has seen move from specific (discovery of e-prints) to generic (sharing descriptions of any resources)
- based on HTTP and XML – Web-friendly
- allows client to say ‘give me some or all of your records’ where ‘some’ is based on
  - datestamps, sets, metadata formats

Summary (2)

- mandates simple DC as record format but extensible to any format encoded in XML
- OAI-PMH is **not** a search protocol
- metadata and full-text typically made freely available – but not a requirement
  - OAI-PMH can be used between closed groups
- access-control and compression mechanisms based on underlying HTTP protocol
- simple protocol allows easy deployment
- systems can be combined in variety of ways

Important resources

- OAI Web site:
  http://ww.openarchives.org/
- OAI-PMH specification:
  http://www.openarchives.org/OAI/openarchivesprotocol.html
- Implementation guidelines:
  http://www.openarchives.org/OAI/2.0/guidelines.htm
- Discussion lists:
  http://www.openarchives.org/mailman/listinfo/oai-general
  http://oaisrv.nsdl.cornell.edu/mailman/listinfo/oai-implementers
- Repository explorer:
  http://oai.dlib.vt.edu/cgi-bin/Explorer/oai2.0/testoai
- Tools:
  http://oai.dlib.vt.edu/cgi-bin/Explorer/oai2.0/testoai

Agenda

Part I - History and Overview

Part II - OAI Serviceprovider - Examples

Part III - Technical Introduction

Part IV - Implementation Issues

Part V - Different Metadata Formats

Tutorial
Open Archive Initiative

Part II
OAI Service Provider - Examples

Service Provider Examples

- Citation Indexing
  http://icite.sissa.it
- Search Engine
  http://arc.cs.odu.edu/
- Printing on demand service
  http://www.proprint-service.de
- Value added Search Engine
  http://www.myoai.com

Agenda

Part I - History and Overview

Part II - OAI Serviceprovider - Examples

Part III - Technical Introduction

Part IV - Implementation Issues

Part V - Different Metadata Formats

Tutorial
Open Archive Initiative

Part III
Technical Introduction

What is an „Open Archive“

- Any WWW-based system that can be accessed through the well-defined interface of the Open Archives Protocol for Metadata Harvesting.
- Is then known as an OAI-compliant archive
- No implications for:
  - Physical storage of data
  - Cost of data
  - Metadata and data formats
  - Access control to server

Reminder: Harvesting vs. Searching

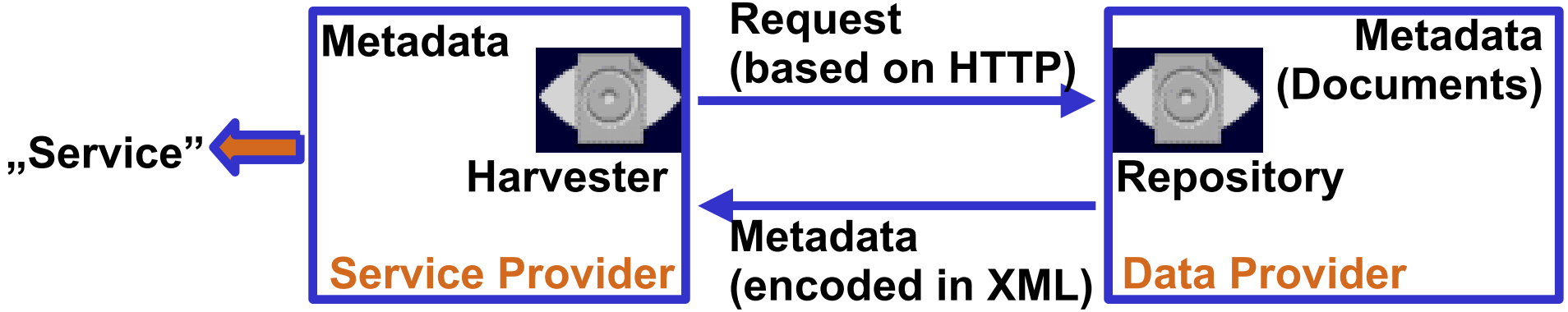
- Competing approaches to interoperability
  - Cross Searching: services are run remotely on remote data (e.g. Federated searching)
  - Harvesting: data/metadata is transferred from the remote source to the destination where the services are located (e.g. Union catalogues)
- Cross Searching requires more effort at each remote source but is easier for the local system and vice versa for harvesting
- OAI actually bases on harvesting

Metadata vs. Data

- Data refers to digital objects or digital representations of objects
- Metadata is information about the objects (e.g. title, author, etc.)
- OAI focuses on metadata, with the implicit understanding that metadata usually contains useful links to the source digital objects

The Open Archives Initiative (OAI)

- Main ideas
  - world-wide consolidation of scholarly archives
  - free access on the archives (at least: metadata)
  - consistent interfaces for archives and service provider
  - low barrier protocol / effortless implementation
  - based on existing standards (e.g. HTTP, XML, DC)
- Basic functioning



Requirements of the Protocol

A communication protocol should …

- be in machine readable format
- encoded in a strict format, which can be validated
  - character encoding
  - metadata encoding
- support different content models
  - metadata formats
- use existing technologies (HTTP, XML, DC)
  - easy to implement
  - easy to adjust

Data and Service Provider

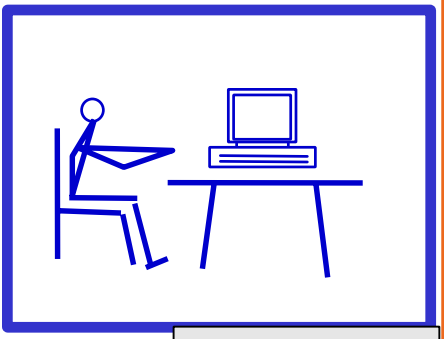
- Data Providers refer to entities who possess data/metadata and are willing to share this with others (internally or externally) via well-defined OAI protocols (e.g. database servers)
- Service Providers are entities who harvest data from Data Providers in order to provide higher-level services to users (e.g. search engines)
- OAI uses these denotations for its client/server model (data=server, service=client)

OAI: General Assumptions

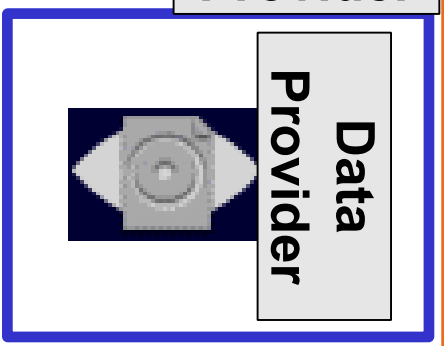
OAI-PMH defines two groups of ‘participants’:

- Data Providers (Open Archives, Repositories)
  - normally: free access of metadata
  - not necessarily: free access to full texts / resources
  - easy to implement, low barriers
- Service Providers
  - use OAI interfaces of the Data Providers
  - harvest and store metadata (no live requests!)
  - may select certain subsets from Data Providers (set hierarchy, date stamp)
  - may enrich metadata
  - offer (value-added) service on the basis of the metadata

OAI-PMH: Structure Model

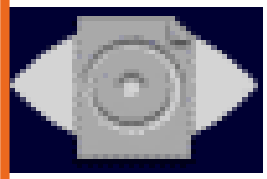


Service Provider



Requests:

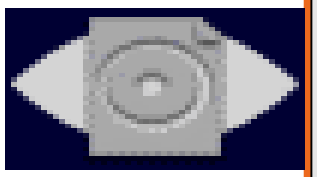
- Identify
- ListMetadataformats
- ListSets
- ListIdentifiers
- ListRecords
- GetRecord

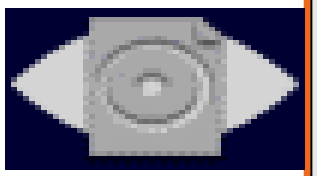


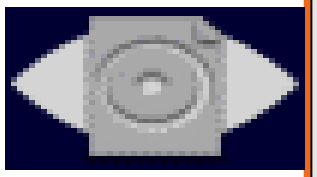
Harvester

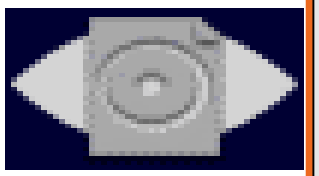
Responses:

- General information
- Metadata formats
- Set structure
- Record identifier
- Metadata









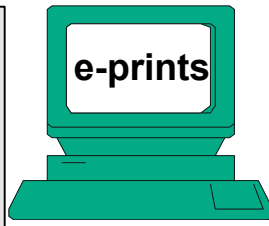
Data Provider

Data Provider

Data Provider

Data Provider

Data Provider



OAI-PMH: Protocol Overview

- Protocol based on HTTP
  - request arguments as GET or POST parameters
  - six request types
  - e.g. **http://archive.org?verb=ListRecords&metadataformat=oai_dc&from=2002-11-01**
  - responses are encoded in XML syntax
  - supports any metadata format (at least: Dublin Core)
  - logical set hierarchy (definition: data providers)
  - datestamps (last change of metadata set)
  - error messages
  - flow control

Protocol Details: Definitions

- **Harvester**
  - client application issuing OAI-PMH requests
- **Repository**
  - network accessible server, able to process OAI-PMH requests correctly
- **Resource**
  - object the metadata is “about”, nature of resources is not defined in the OAI-PMH
- **Item**
  - component of a repository from which metadata about a resource can be disseminated
  - has a unique identifier

Protocol Details: Definitions (2)

- **Item**
  - component of a repository from which metadata about a resource can be disseminated
  - has a unique identifier
- **Record**
  - metadata in a specific metadata format
- **Identifier**
  - unique key for an item in a repository
- **Set**
  - optional construct for grouping items in a repository

Protocol Details: Definitions (3)

resource

item = identifier

Metadata about *David*

item

Dublin Core metadata

MARC metadata

SPECTRUM metadata

record

What is a „Record“?

- refers to an independent XML structure that may be associated with digital or physical objects
- is usually associated with metadata, not data
- is the representation of an item in a specific metadata format
- OAI advocates harvesting of records, which contain metadata and additional fields to support the harvesting operation

Uniqueness and Persistence

- Each record must be uniquely addressable by a distinct identifier

  (**`identifier + metadataPrefix`**)

- Each metadata entity should ideally be persistent to guarantee that service providers can always refer back to the source.

Protocol Details: Records

- metadata of a resource in a specific format
- consists of three parts
  - **header (mandatory)**
    - identifier (1)
    - datestamp (1)
    - setSpec elements (*)
    - status attribute for deleted item (?)
  - **metadata (mandatory)**
    - XML encoded metadata with root tag, namespace
    - repositories must support Dublin Core
  - **about (optional)**
    - rights statements
    - provenance statements

1 … occurs exactly once

* … optional, can occur more than once

? … occurs zero times or exactly once

Example: OAI Record

(NOTE: Schema and Namespaces have been removed for simplicity)

```
<record>
  <header>
    <identifier>oai:YOOWE.de:1</identifier>
    <datestamp>2004-02-12</datestamp>
    <setSpec>tutorial</setSpec>
  </header>
  <metadata>
    <oai_dc>
      <title>OAI-PMH Implementation</tritle>
      <creator>Uwe Müller</creator>
      <language>eng</language>
    </oai_dc>
  </metadata>
  <about>
    <rights>You are free to reuse this</rights>
  </about>
</record>
```

Date stamps & Harvesting

- date stamp: date of last modification of the **metadata**
- mandatory characteristic of every item
- two possible granularities:
  - YYYY-MM-DD
  - YYYY-MM-DDThh:mm:ssZ
- function: information on metadata, selective harvesting (**`from`** and **`until`** arguments)
- applications: incremental update mechanisms
- modification, creating, deletion
- deletion: three support levels
  - no, persistent, transient

Metadata Schemes

- OAI-PMH supports dissemination of multiple metadata formats from a repository
- properties of metadata formats
  - id string to specify the format (metadataPrefix)
  - metadata schema URL (XML schema to test validity)
  - XML namespace URI (global identifier for metadata format)
- repositories must be able to disseminate at least unqualified Dublin Core
- **arbitrary metadata formats** can be defined and transported via the OAI-PMH
- returned metadata must comply with XML schema and namespace specification

Sets

- protocol mechanism to allow for harvesting of sub-collections
- no well-defined semantics – depends completely on local data providers
- May be defined by arrangement between data providers and service providers
- applications:
  subject gateways, dissertation search engine, …
- examples (Germany, see http://www.dini.de)
  - publication types (thesis, article, …)
  - document types (text, audio, image, …)
  - content sets, regarding DNB (medicine, biology, …)

OAI-PMH Request Format

- requests must be submitted using the **GET** or **POST** methods of HTTP
- repositories must support both methods
- at least one key=value pair: verb=[RequestType]
- additional key=value pairs depend on request type
- example for **GET** request:
  http://archive.org/oai?
  verb=ListRecords&metadataPrefix=oai_dc
- encoding of special characters
  e.g. “**:**” (host port separator) becomes “**%3A**”

OAI-PMH Response Format

- formatted as HTTP responses
- content type must be text/xml
- status codes (distinguished from OAI-PMH errors)
  e.g. 302 (redirect), 503 (service not available)
- response format: well formed XML with markup:
  1. XML declaration
     (**`<?xml version="1.0" encoding="UTF-8" ?>`**)
  2. root element named **`OAI-PMH`** with three attributes
     (**`xmlns`**, **`xmlns:xsi`**, **`xsi:schemaLocation`**)
  3. three child elements
     1. **`responseDate`** (UTC datetime)
     2. **`request`** (request that generated this response)
     3. a) **`error`** (in case of an error or exception condition)
        b) element with the name of the OAI-PMH request

Example Response (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2003-05-24T10:23:21Z</responseDate>
 <request verb="GetRecord" metadataPrefix="oai_dc"
  identifier="oai:ex-dp:93">http://example-data-
    provider/oai-interface.php</request>
 <GetRecord>
  <record>
  <header>
   <identifier>oai:ex-dp:93</identifier>
   <datestamp>2003-05-01T00:00:00Z</datestamp>
  </header>
```

Example Response (2)

```
 <metadata>
   <oai_dc:dc
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
 oai_dc/  http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
    <dc:title>Thoughts about OAI</dc:title>
    <dc:date>2003-04-22</dc:date>
    <dc:identifier>http://example-data-provider/oai.pdf
    </dc:identifer>  <dc:language>eng</dc:language>
   </oai_dc:dc>
  </metadata>
 </record>
</GetRecord>
</OAI-PMH>
```

Flow Control

- flow control on two protocol levels
  - HTTP (503, retry-after)
  - OAI-PMH, Resumption-Token
- HTTP “retry-after” mechanism can be used in order to delay requests of clients
- resumption tokens are used to return parts (incomplete lists) of the result.
- client receive a token which can be used to issue another request – in order to receive further parts of the result

Flow Control (2)

- four of the request types return a list of entries
- three of them may reply ‘large’ lists
- OAI-PMH supports partitioning
- decision on partitioning: repository
- response to a request includes
  - incomplete list
  - resumption token
    + expiration date, size of complete list, cursor (optional)
- new request with same request type
  - resumption token as parameter
  - all other parameters omitted!
- response includes
  - next (maybe last) section of the list
  - resumption token (empty if last section of list enclosed)

Flow Control (3) – Example

Errors and Exceptions

- repositories must indicate OAI-PMH errors
- inclusion of one or more **`error`** elements
- defined error identifiers
  - **`badArgument`**
  - **`badResumptionToken`**
  - **`badVerb`**
  - **`cannotDisseminateFormat`**
  - **`idDoesNotExist`**
  - **`noRecordsMatch`**
  - **`noMetaDataFormats`**
  - **`noSetHierarchy`**

Request Types

- six different request types
  1. Identify
  2. ListMetadataFormats
  3. ListSets
  4. ListIdentifiers
  5. ListRecords
  6. GetRecord
- harvester has not to use all types
- repository must implement all types
- required and optional arguments
- depend on request types

Request: Identify

- Function
  - general information about archive
- Parameter
  - none
- Example URL
  - **http://physnet.de/oai/oai2.php?verb=Identify**
- Errors/Exceptions
  - **`badArgument`** e.g. **physnet.de/oai/oai2.php?verb=Identify&set=biology**

Request: Identify (2)

Request:

http://physnet.uni-oldenburg.de/oai/oai2.php?verb=Identify

Response (1):

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2003-05-24T10:27:14Z</responseDate>
 <request verb="Identify">
    http://physnet.uni-oldenburg.de/oai/oai2.php</request>
 <Identify>
  <repositoryName>Physnet, GERMANY, Document Server
  </repositoryName>
  <baseURL>http://physnet.uni-oldenburg.de/oai/oai2.php
  </baseURL>
```

Request: Identify (3)

Response (2):

```
<protocolVersion>2.0</protocolVersion>
<adminEmail>mailto:stamer@uni-oldenburg.de</adminEmail>
<earliestDatestamp>2000-01-01</earliestDatestamp>
<deletedRecord>no</deletedRecord>
<granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
<description>
 <friends xsi:schemaLocation=
  “http://www.openarchives.org/OAI/2.0/friends/
   http://www.openarchives.org/OAI/2.0/friends.xsd”>
  <baseURL>http://uni-d.de:8080/cgi-oai/oai.pl</baseURL>
  <baseURL>http://edoc.hu-berlin.de/OAI2.0</baseURL>
  <baseURL>http://naca.larc.nasa.gov/oai2.0/</baseURL>
 </friends>
</description>
</Identify>
</OAI-PMH>
```

Request: Identify (3)

Response format

Element	*Example*	#
repositoryName	My Archive	1
baseURL	http://archive.org/oai	1
protocolVersion	2.0	1
earliestDatestamp	1999-01-01	1
deleteRecords	no, transient, persistent	1
granularity	YYYY-MM-DD, YYYY-MM-DDThh:mm:ssZ	1
adminEmail	oai-admin@archive.org	+
compression	deflate, compress, …	*
description	oai-identifier, eprints, friends, …	*

1 … occurs exactly once, + …occurs at least once,
* … optional, can occur more than once

Request: ListMetadataFormats

- Function
  - list metadata formats, which are supported by archive, as well as their Schema Locations and Namespaces
- Parameter
  - identifier – for a specific record (optional)
- Example URL
  - **http://physnet.de/oai/oai2.php?**
    verb=ListMetadataFormats
- Errors/Exceptions
  - **`badArgument`**
  - **`idDoesNotExist`** e.g.

    archive.org/oai-script?
    verb=ListMetadataFormats&
    identifier=really-wrong-identifier
  - **`noMetadataFormats`**

Request: ListMetadataFormats (2)

Request:

http://physnet.uni-oldenburg.de/oai/oai2.php?verb=ListMetadataFormats

Response (1):

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2003-05-24T10:29:29Z</responseDate>
  <request verb="ListMetadataFormats">
     http://physnet.uni-oldenburg.de/oai/oai2.php
  </request>
```

Request: ListMetadataFormats (3)

Request:

http://physnet.uni-oldenburg.de/oai/oai2.php?verb=ListMetadataFormats

Response (2):

```
<ListMetadataFormats>
  <metadataFormat>
    <metadataPrefix>oai_dc</metadataPrefix>
    <schema>
        http://www.openarchives.org/OAI/2.0/oai_dc.xsd
    </schema>
    <metadataNamespace>
        http://www.openarchives.org/OAI/2.0/oai_dc
    </metadataNamespace>
  </metadataFormat>
</ListMetadataFormats>
</OAI-PMH>
```

Request: ListSets

- Function
  - hierarchical listing of Sets in which records have been organized
- Parameter
  - none
- Example URL
  - **http://physnet.de/oai/oai2.php?verb=ListSets**
- Errors/Exceptions
  - **`badArgument`**
  - **`badResumptionToken`** e.g. **archive.org/oai-script?verb=ListSets&resumptionToken=any-wrong-token**
  - **`noSetHierarchy`**

Request: ListIdentifiers

- Function
  - retrieve headers of all Records, which comply to parameters
- Parameter
  - **`from`** – Startdate (optional)
  - **`until`** – Enddate (optional)
  - **`set`** – Set of which to be harvested (optional)
  - **`metadataPrefix`** – metadata format, for which Identifier should be listed (required)
  - **`resumptionToken`** – flow control (exclusive)
- Example URL
  - **http://physnet.de/oai/oai2.php?verb=ListIdentifiers&metadataPrefix=oai_dc**

Request: ListIdentifiers (2)

- Errors/Exceptions
  - **`badArgument`**, e.g..
    …**&from=2002-12-01T13:45:00**
    (here: wrong granularity)
  - **`badResumptionToken`**
  - **`cannotDisseminateFormat`**
  - **`noRecordsMatch`**
  - **`noSetHierarchy`**

Request: ListRecords

- Function
  - retrieve multiple Records
- Parameter
  - **`from`** – Startdate (O)
  - **`until`** – Enddate (O)
  - **`set`** – Set from which to be harvested (O)
  - **`metadataPrefix`** – metadata format (R)
  - **`resumptionToken`** – flow control (X)
- Example URL
  - **http://physnet.de/oai/oai2.php?**
    verb=ListRecords&
    metadataPrefix=oai_dc&from=2001-01-01

Request: ListRecords (2)

- Errors/Exceptions
  - **`badArgument`**
  - **`badResumptionToken`**
  - **`cannotDisseminateFormat`**
  - **`noRecordsMatch`**
  - **`noSetHierarchy`**

Request: ListRecords (3)

Response (1):

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2003-05-24T10:23:21Z</responseDate>
 <request verb="ListRecords" metadataPrefix="oai_dc">
    http://physnet.uni-oldenburg.de/oai/oai2.php</request>
 <ListRecords>
  <record>
   <header>
    <identifier>oai:physdoc:5987</identifier>
    <datestamp>2002-01-25T00:00:00Z</datestamp>
   </header>
```

Request: ListRecords (4)

Response (2):

```
 <metadata>
  <oai_dc:dc xmlns:oai_dc=
“http://www.openarchives.org/OAI/2.0/oai_dc/”
xmlns:dc=“http://purl.org/dc/elements/1.1/”
xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=“http://www.openarchives.org/OAI/2.0/
oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd”>
   <dc:title>Pole de Calcul Parallele</dc:title>
   <dc:date>2003-01-05</dc:date><dc:identifier>
    http://physnet.uni-oldenburg/pole.pdf</dc:identifer>
  </oai_dc:dc>
 </metadata>
</record>
[... more records ...]
 </ListRecords>
</OAI-PMH>
```

Request: GetRecord

- Function
  - return single Record
- Parameter
  - **`identifier`** – unique ID for Record (required)
  - **`metadataPrefix`** – metadata format (required)
- Example URL
  - **http://physnet.de/oai/oai2.php?**
    verb=GetRecord&identifier=oai:test:123&
    metadataPrefix=oai_dc
- Errors/Exceptions
  - **`badArgument`**
  - **`cannotDisseminateFormat`**
  - **`idDoesNotExist`**

Example: Date Ranges

Request:

http://rocky.dlib.vt.edu/~jcdlpix/cgi-bin/OAI2.0/beta2/jcdl/oai.pl?
verb=ListIdentifiers&from=2001-06-26&until=2001-06-26&
metadataPrefix=oai_dc

Response (1):

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2002-05-26T19:41:16Z</responseDate>
 <request verb="ListIdentifers" from="2001-06-26"
          until="2001-06-26" metadataPrefix="oai_dc">
    http://rocky.dlib.vt.edu/~jcdlpix/cgi-
      bin/OAI2.0/beta2/jcdl/oai.pl
 </request>
```

Example: Date Ranges (2)

Response (2):

```
<ListIdentifers>
  <header>
      <identifier>oai:JCDLPICS:200102dlb1</identifier>
      <datestamp>2001-06-26</datestamp>
      <setSpec>200102dlb</setSpec>
    </header>
    <header>
      <identifier>oai:JCDLPICS:200102dlb2</identifier>
      <datestamp>2001-06-26</datestamp>
      <setSpec>200102dlb</setSpec>
    </header>

[... more headers ...]

  </ListIdentifiers>
</OAI-PMH>
```

Agenda

Part I - History and Overview

Part II - OAI Serviceprovider - Examples

Part III - Technical Introduction

Part IV - Implementation Issues

Part V - Different Metadata Formats

Tutorial
Open Archive Initiative

Part IV
Implementation of
Data and Service Provider

General: First Questions

Data Provider

- What kind of data do I want to provide?
- (To which Service Providers will I offer my data?)

Service Provider

- What kind of service do I want to provide?
- From whom (Data Providers) do I want to collect data?
- What kind of metadata format do I want (need) to support?

Data Provider & Service Provider

- Do I need to have agreements on certain aspects?
- Metadata formats, Sets ...

Metadata Mappings

- Data Provider must **map** its internal metadata to format, which it offers through OAI Interface.
- Unqualified Dublin Core is mandatory as least common denominator
  - http://dublincore.org/
  - Dublin Core Metadata Element Set has 15 Elements
  - Elements are optional, and can be repeated
  - Normally a Link to Resource is provided in the **<identifier>** Tag
- Source metadata formats are recommended
- Metadata formats of your own community are recommended

Organisation

- required: unqualified Dublin Core
- special subjects / communities: other metadata specifications may be required
  - describe resources in a specialised way
  - definition of an XML schema (publicly available for validation)
- define set hierarchy
  - sensible partitioning for selective harvesting
  - agreement between data providers and between data and service providers

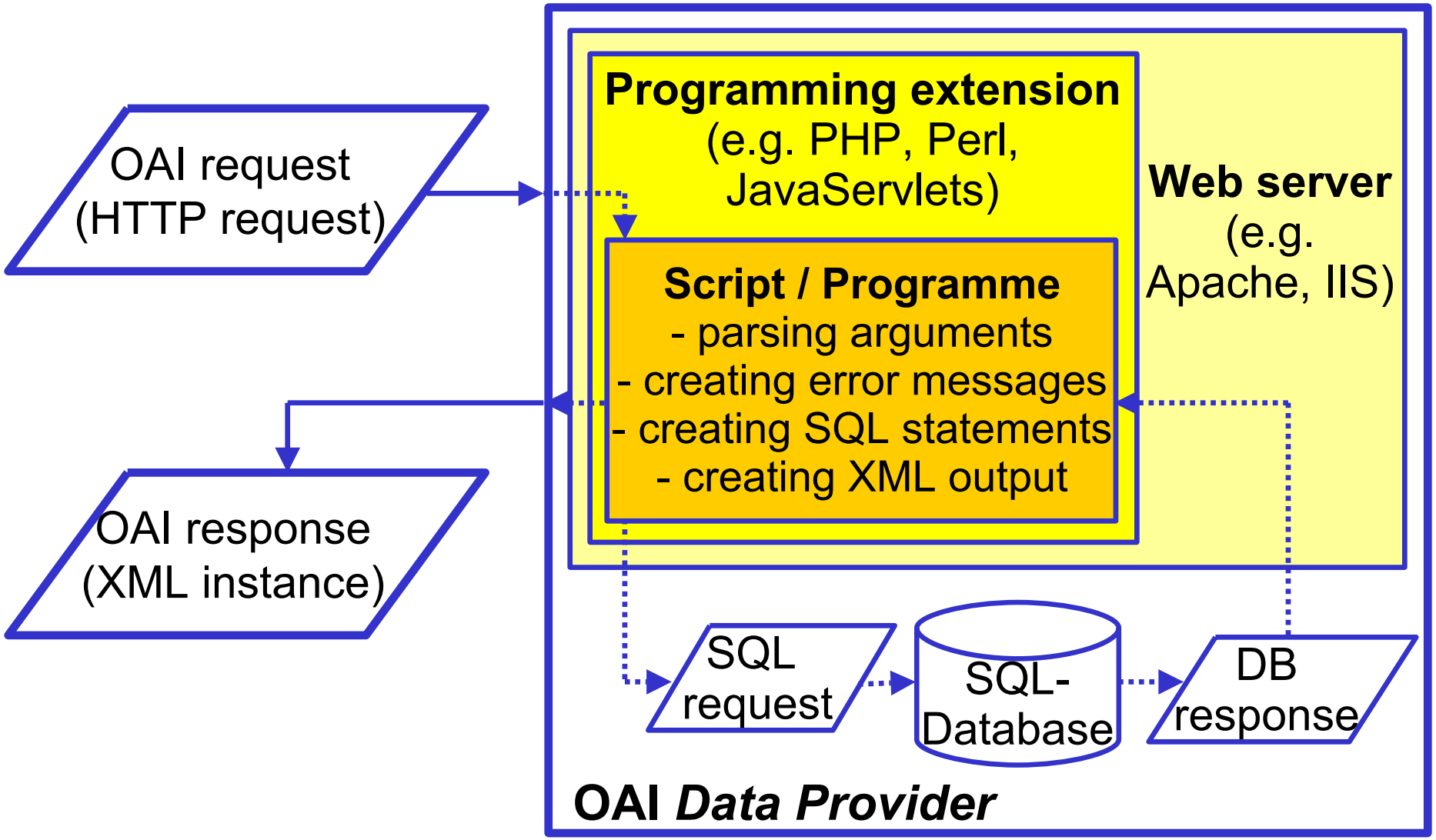
Server Technology

- WWW Server
- Protocol may be implemented in arbitrary form, e.g.
  - CGI script (Perl, C++, Java)
  - Java servlet
  - PHP
- Metadata (e.g. database) access necessary
- See http://www.openarchives.org for list of software.

Metadata Sources

- Database in proprietary format, can be either SQL or XML databases
- Metadata collections in well-defined format(s)
  - e.g. files on disk
- Metadata can be extracted dynamically or statically from data
  - to serve XML, no storage of XML necessary
  - data from SQL database can be easily converted to XML on-the-fly

Data Provider: Architecture



Datestamps

- Needed for every record to support incremental harvesting
- Must be updated for every addition/modification/deletion to ensure changes are correctly propagated
- Different from dates within the metadata – this date is used only for harvesting
- Can be either YYYY-MM-DD or YYYY-MM-DDThh:mm:ssZ (must be GMT timezone)

Unique Identifier

- Each record must have a unique identifier
- Identifiers must be valid URIs
- Example:
  - oai:<archiveId>:<recordId>
  - oai:etd.vt.edu:etd-1234567890
- Each identifier must resolve to a single record and always to the same record (for a given metadata format)

Deletions

- Archives may keep track of deleted records, by identifier and datestamp
- All protocol result sets can indicate deleted records
- If deletions are being tracked, this information must be stored indefinitely so as to correctly propagate to service providers with varying harvesting schedules

Required Tools

- for new collections have a look at existing software
  - Eprints
  - Dspace
  - ETD software from VT
- to make existing collections OAI compliant
  - use web scripts
  - look for existing tools on
    - http://www.openarchives.org
    - http://edoc.hu-berlin.de/oai
  - open source, easy to adapt to local needs.

Data Provider: General Structure

- Argument Parser
  - validates OAI requests
- Error Generator
  - creates XML responses with encoded error messages
- Database Query / Local Metadata Extraction
  - retrieves metadata from repository
  - according to the required metadata format
- XML Generator / Response Creation
  - creates XML responses with encoded metadata information
- Flow Control
  - realises incomplete list sequences for ‘larger’ repositories
  - uses resumption token as mechanism

Data Provider: Resumption Token

- should be implemented for “large” lists
- initiated by data provider
- store parameters (**`set`**, **`from`**, …) and number of delivered records
- properties
  - expiration: expirationDate (optional)
  - completeListSize (optional)
  - already delivered records: cursor (optional)
  - recovery from network errors (possibility to re-issue most recent resumption token)
- problem: database changes
  - two possible solutions
    - duplicate data in a “request table”
    - store date of first request with the other parameters use like additional until argument

Resumption Token (2)

Request:

edoc.hu-berlin.de/OAI-2.0?verb=ListRecords&metadataPrefix=oai_dc

Response (1):

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
   http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2003-05-24T11:41:16Z</responseDate>
  <request verb="ListRecords" metadataPrefix="oai_dc">
    http://edoc.hu-berlin.de/OAI-2.0</request>
  <ListRecords>
    <records>

[... header and metadata information ...]

    </recrods>
```

Resumption Token (3)

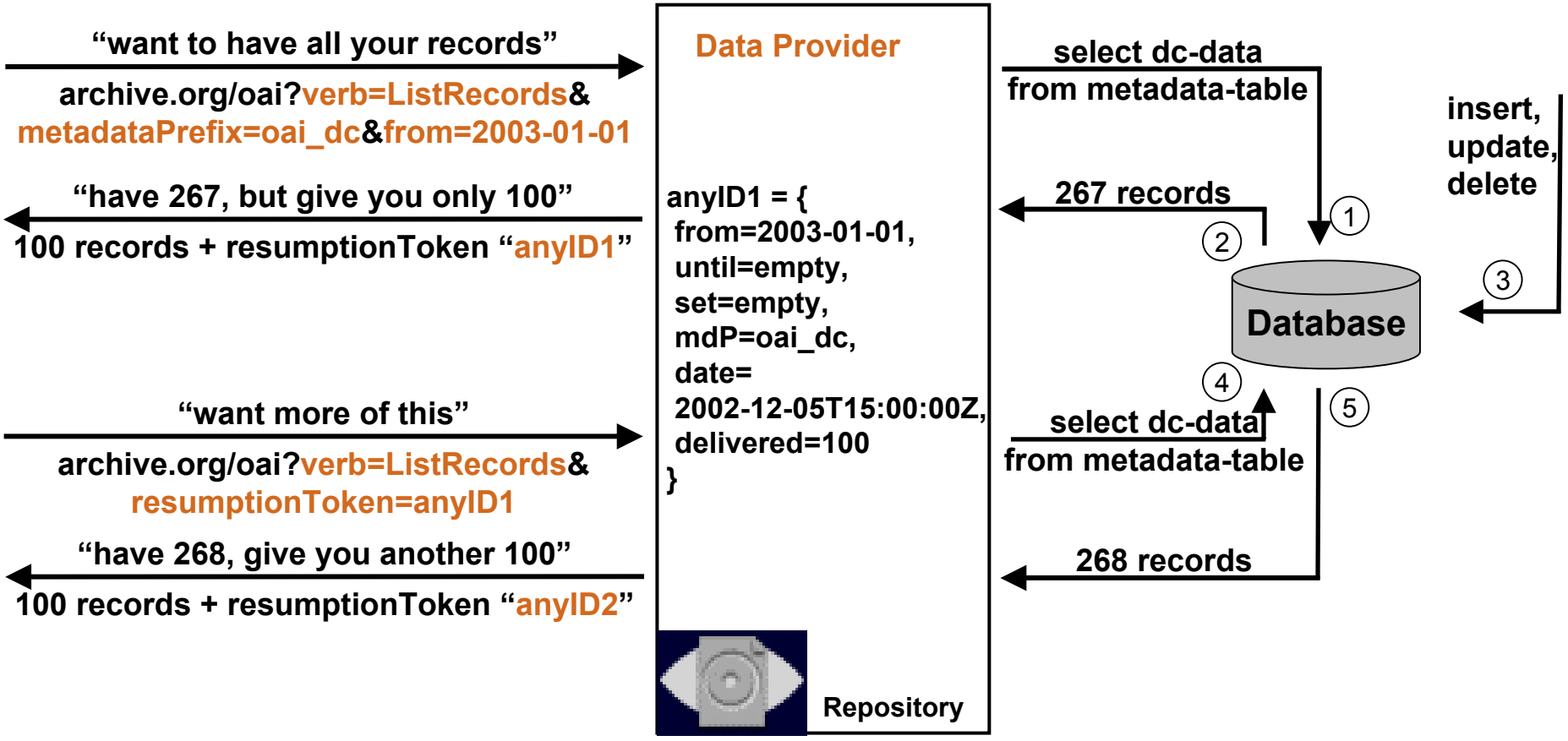
Request:

edoc.hu-berlin.de/OAI-2.0?verb=ListRecords&metadataPrefix=oai_dc

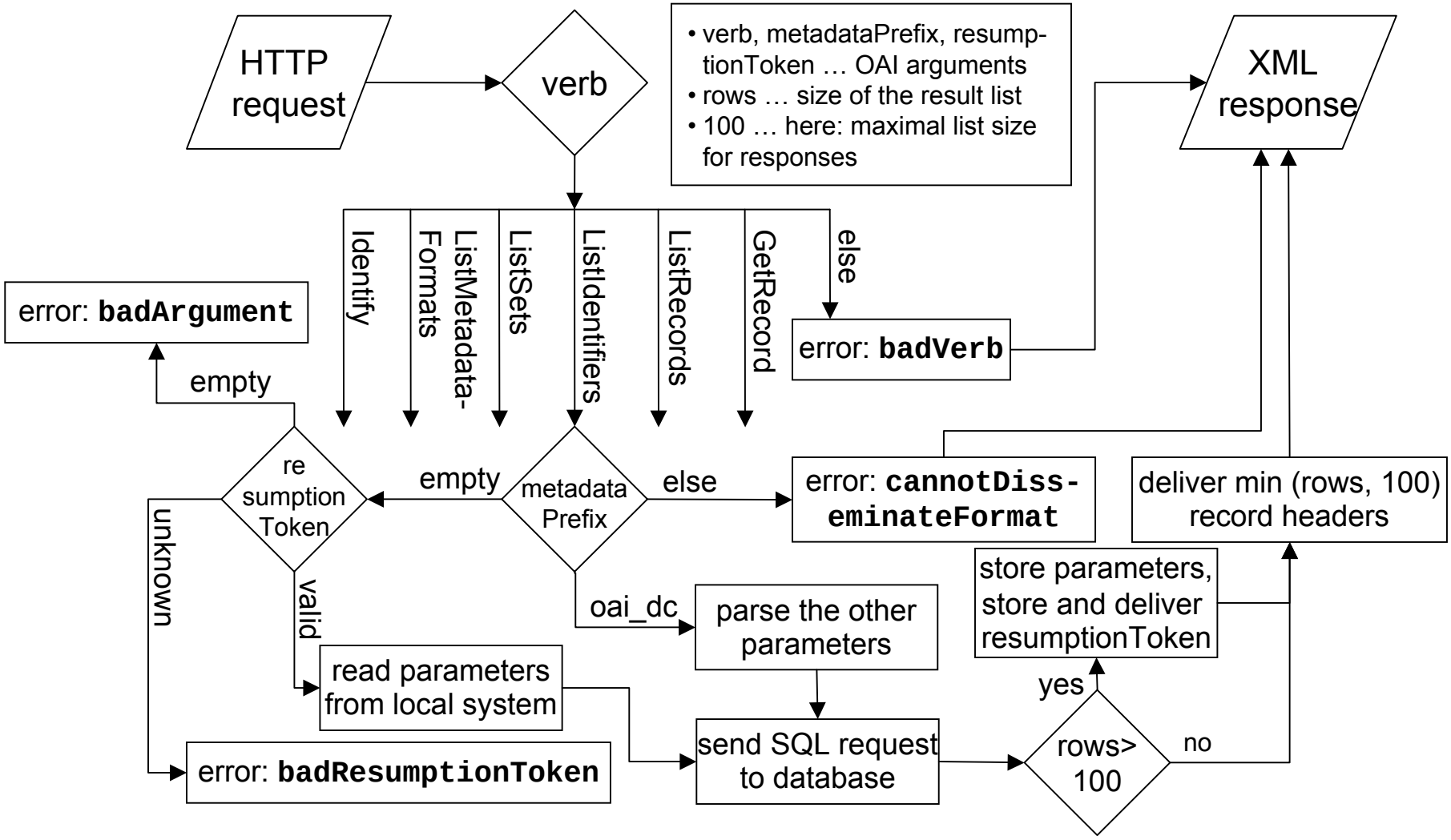
Response (2):

```
    <records>
[... header and metadata information ...]
    </recrods>
[... more records ...]
    <resumptionToken expirationDate="2003-05-26T00:00:00Z"
                     completeListSite="319"
                     cursor="0">312898978423
    </resumptionToken>
  </ListRecords>
</OAI-PMH>
```

Resumption Token (4)

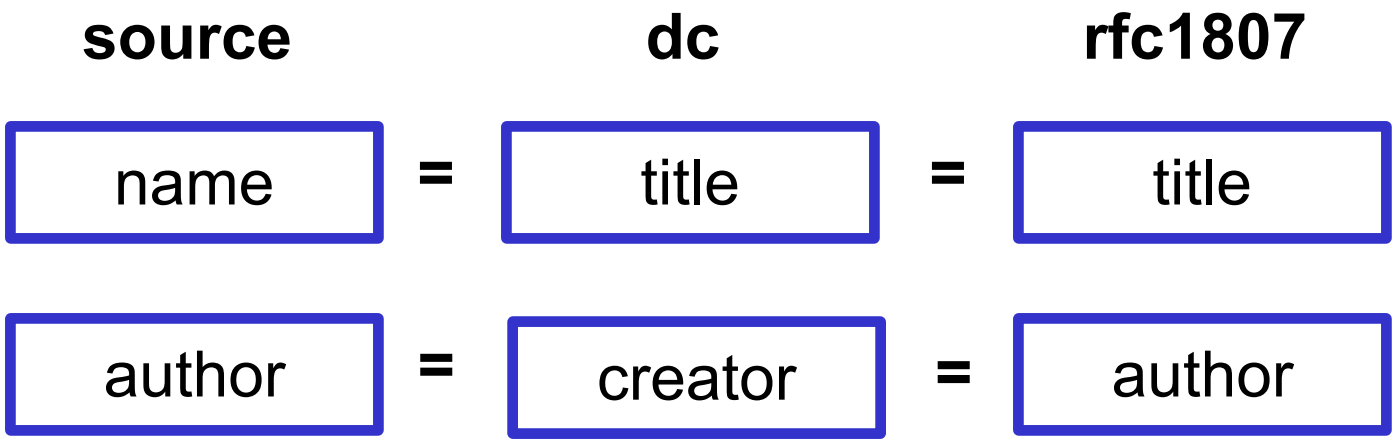


Data Provider: Example Flow Chart



Metadata Creation

- Approaches:
  - Map from source to each metadata format
  - Use crosswalks (maybe XSLT) to generate additional formats



Data Provider: Data Representation

- use recommended data representation
  - dates
    - ■ 2002-12-05
    - ✘ 2002-xx-xx, 2002, 05.12.2002
  - language code
    - ■ eng, ger, ...
    - ✘ en, de, english, german
- multi values: use own XML element for each entity
  - author
    - ■ <dc:creator>Smith, Adam</dc:creator> <dc:creator>Nash, John</dc:creator>
    - ✘ <dc:creator>Smith, Adam; Nash, John </dc:creator>

Encoding data for XML

- Special XML Characters must be escaped
  - <>&
- Convert to UTF-8 (Unicode)
- Convert entities
- Remove unneccessary spaces
- Convert CR/LF for paragraphs
- URLs
  - /?#=&:;+ must be encoded as escape sequence

Data Provider: Compression

- method to reduce traffic and enhance performance
- optional for both sides: data and service providers
- handled on HTTP level
- harvesters may include an Accept-Encoding header in their requests –specifying preferences
- harvesters without Accept-Encoding header always receive uncompressed data
- repositories must support HTTP identity encoding
- repositories should specify supported encodings by including compression elements in the identify response

Error Handling

- All protocol errors are in XML format
  - **`badVerb`**

    illegal verb requested
  - **`badArgument`**

    illegal parameter values or combinations
  - **`badResumptionToken`, `cannotDisseminateFormat`, `idDoesNotExist`**

    parameters are in right format but are not legal under current conditions
  - **`noRecordsMatch`, `noMetadataFormats`, `noSetHierarchy`**

    empty response exception

Error Handling: Example

Request:

http://physnet.uni-oldenburg.de/oai/oai2.php?verb=IllegalVerb

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
   http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2003-05-24T11:53:30Z</responseDate>
  <request>http://physnet.uni-
           oldenburg.de/oai/oai2.php</request>
  <error code="badVerb">The verb 'IllegalVerb'
           provided in the request is illegal</error>
</OAI-PMH>
```

Common Problems

- No unique identifiers
- No date stamps
- Incomplete information in database
- New metadata format
- XML responses not validating

No Unique Identifiers

- Create an independent identifier mapping
- Use row numbers for a database
- Use filenames for data in files
- Use a hash from other fields (poor solution!)
  - e.g. calculate identifier as a hash value of the string created by concatenating the values of **author** + **year** + **first word in title**

No Datestamps

- Ignore the datestamp parameters and stamp all records with the current date
- Create a date table with the current date for all old entries and update dates for new entries
- Most Important: Any harvesting algorithm that is interoperably stable for an archive with real dates should be stable for an archive with synthesized dates

Incomplete Information

- Synthesize metadata fields based on a priori knowledge of the data
  - Example: publisher and language may be hard-coded for many archives
  - Omit fields that cannot be filled in correctly – better to have less information than incorrect information !

New Metadata Format

- Find the description, namespace and formal name of the standard
- Find an XML Schema description of the data format
  - If none exists, write one (consult other OAI people for assistance)
  - Create the mapping and test that it passes XML schema validation

Not Validating XML

- Check namespaces and schema
- Use Repository Explorer in non-validating mode to check structure of XML, without looking at namespaces or schemata
- Validate schema by itself if it is non-standard
- Look at XML produced by other repositories
- Watch out for common character encoding issues (iso8859-1 → utf-8)

Tools for Testing

- Repository Explorer
  - Interactive Browsing
  - Testing of parameters
  - Multiple views of data
  - Multilingual support
  - Automatic test suite
- OAI Registry
- XML Schema Validator

Service Provider: Requirements

- internet connected server
- database system (relational or XML)
- programming environment
  - can issue HTTP requests to web servers
  - can issue database requests
  - XML parser

Service Provider: Structure (1)

Archive Management

- selection of archives to be harvested
- enter entries manually or
- automatically add / remove archives using the official registry

Request Component

- creates HTTP requests and sends them to OAI archives (data provider)
- demands metadata using the allowed verbs of the OAI-PMH
- possibly selective harvesting (set parameter)

Service Provider: Structure (2)

Scheduler

- realises timed and regular retrieval of the associated archives
- simplest case: manual initiation of the jobs
- else: e.g. cron job …

Flow Control

- resumption token: partitioning of the result list into incomplete sections – anew request to retrieve more results
- HTTP error 503 (service not available) – analysis of response to extract “retry-after” period

Service Provider: Structure (3)

Update Mechanism

- realises consolidation of metadata which have been harvested earlier (merge old and new data)
- easiest case: always delete all ‘old’ metadata of an archive before harvesting it
- reasonable: incremental update (from parameter) – insert new metadata and overwrite changed / deleted metadata (assignment using the unique identifiers)

XML Parser

- analyses the responses received from the archives
- validation: using the XML schema
- transforms the metadata encoded in XML into the internal data structure

Service Provider: Structure (4)

Normaliser and Mapper

- transforms data into a homogenous structure (different metadata formats)
- harmonises representation (e.g. date, author, language code)
- maps / translates different languages

Database

- mapping the XML structure of the metadata into a relational database (multi values …)
- or: use an XML database

Service Provider: Structure (5)

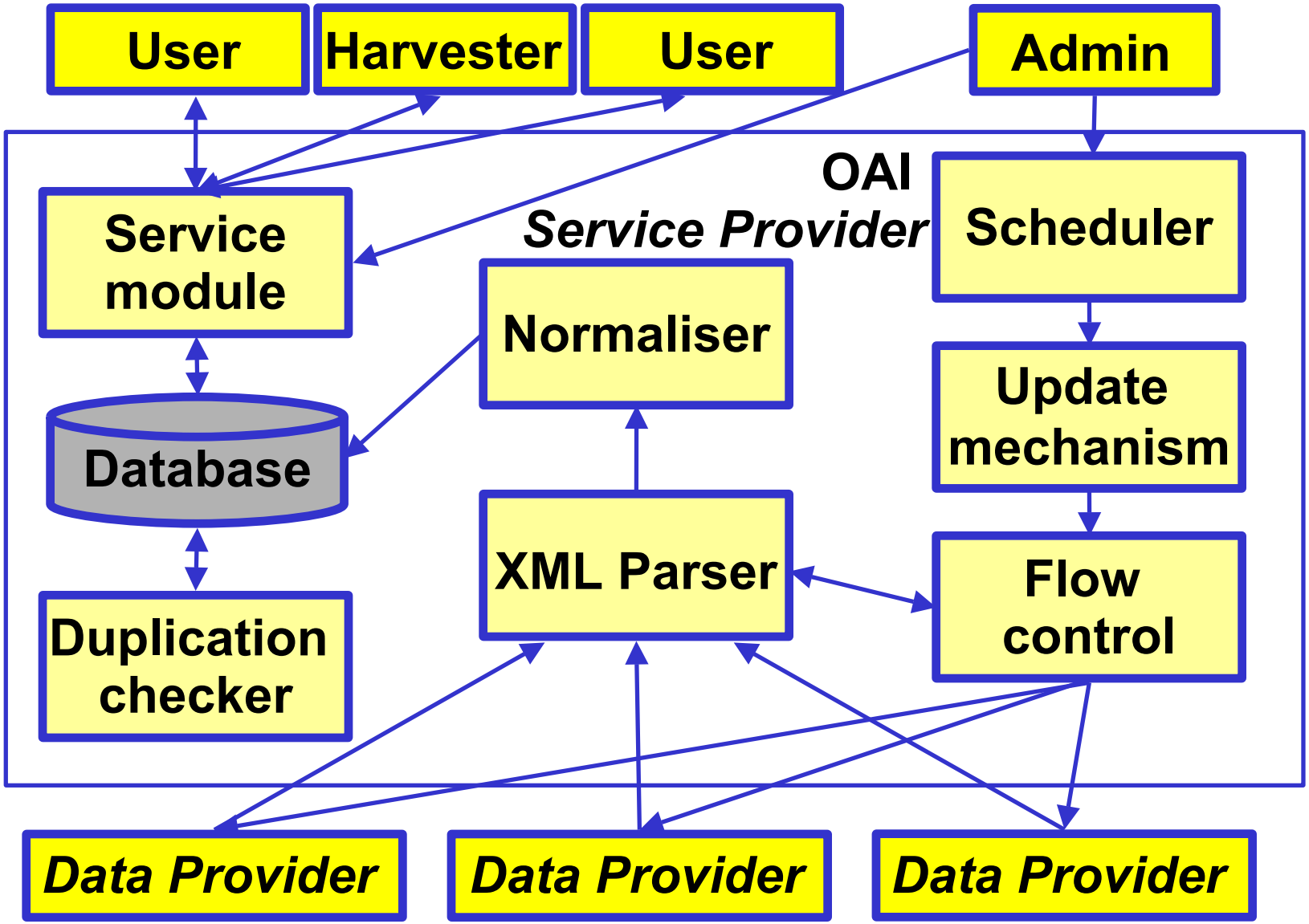
Duplication Checker

- merges identical records from different data providers
- possibility: unique identifier for the item (e.g. URN, …)
- but: often not easily practicable and not risk / error free

Service Module

- provides the actual service to the ‘public’
- basis: harvested and stored records of the associated archives
- uses only local database for requests etc.

Service Provider: Architecture



How to Harvest

- **`Identify`** to get basic information
- **`ListIdentifiers`**, followed by **`ListMetadataFormats`** for each record and then **`GetRecord`** for each id/metadata combination
  - No. of short HTTP requests = 1+n+n x m
    n=no. of identifiers, m=no. of metadata formats
- **`ListRecords`** for each metadata format required
- No. of long HTTP requests = m
  m=no. of metadata formats

Harvest Policies

- Use schedule for harvesting regularly
- Store date when last harvested (before you start)
- Use a two day overlap (or one day if your archive uses proper UTC datestamps)
  - New items may be added for the current day
  - Timezones create up to a day of lag if you ignore them
  - If the source uses correct UTC datestamps and second granularity then only 1 second of overlap is needed!
- Each time a record is encountered, erase previous instances

Intermediate Systems

- Both a data provider and service provider
- All harvested data must have the datestamps updated to the date on which the harvesting was done
- Identifiers retain their original values
- Note: Consistency in the source archive propagates, but so does inconsistency!

Tools

- Check OAI website for sample code
- XML parsers – depending on platform – check W3C
- XML Schema validators
  - Very few available – the reference version works but may not be easy to install
  - Ignore validation if you can trust the source
  - Sample data providers – check the OAI website for a list of conformant public archives

Agenda

Part I - History and Overview

Part II - OAI Serviceprovider - Example

Part III - Technical Introduction

Part IV - Implementation Issues

Part V - Different Metadata Formats

Tutorial
Open Archive Initiative

Part V

Definition and Usage of Different Metadata Formats

The Basics

- OAI-PMH uses XML Schemas
- any metadata format with an XML Schema:
  OK for OAI
- OAI-PMH mandates ‘oai_dc’ schema
- OAI-PMH documentation includes schema for
  - RFC1807 metadata
  - MARC21 metadata (Library of Congress)
  - oai_marc metadata

oai_dc

- Simple unqualified DC schema
- Mandatory ‘Lowest Common Denominator’
- Container schema is OAI specific
- Container schema hosted at OAI Web site
- Imports a generic DCMES schema
- DCMES schema at DCMI Web site

Example Record (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2003-05-24T10:23:21Z</responseDate>
 <request verb="GetRecord" metadataPrefix="oai_dc"
  identifier="oai:ex-dp:93">http://example-data-
    provider/oai-interface.php</request>
 <GetRecord>
  <record>
  <header>
   <identifier>oai:ex-dp:93</identifier>
   <datestamp>2003-05-01T00:00:00Z</datestamp>
  </header>
```

Example Record (2)

```
 <metadata>
   <oai_dc:dc
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
 oai_dc/  http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
    <dc:title>Thoughts about OAI</dc:title>
    <dc:date>2003-04-22</dc:date>
    <dc:identifier>http://example-data-provider/oai.pdf
    </dc:identifer>  <dc:language>eng</dc:language>
   </oai_dc:dc>
  </metadata>
 </record>
</GetRecord>
</OAI-PMH>
```

oai_dc - A Record

three important things to notice:

- namespace for the oai_dc format

  xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"

- namespace for DCMES elements

  xmlns:dc="http://purl.org/dc/elements/1.1/"

- container schema associated with the oai_dc namespace

  xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
  http://www.openarchives.org/OAI/2.0/oai_dc.xsd"

The XML Schemas

- The oai_dc “container schema”
- Imports DCMES schema
- Defines a container element - ‘dc’
- Lists the allowed elements within the ‘dc’ container (defined in DCMES Schema)

Other metadata formats

- oai_dc is a simple format providing baseline interoperability

- It may not be suitable:
  - Not enough (or the required) elements!
  - Not very precise - it is an “unqualified” MES
    (not covered in this talk... Sorry!)
  - Not the metadata format you need i.e. not:
    IMS/IEEE LOM - eLearning metadata
    ODRL - Open Digital Rights Language

oai_dc is ... not enough

Scenario: *print on demand* service

- Needs information on number of pages

Extend the Schema by adding new elements:

- Create a name for new schema
- Create namespaces
- Create the schema for the new elements
- Create ‘container schema’
- Validate your schema / records
- Add to repository’s “ListMetadataFormats”
- Add to repository’s other verbs
- Test it worked and is valid

Step 1: Name your format

- I’m choosing “oai_pod”
- Could be anything you like...

Step 2: Create Namespaces

- We need two namespaces:
  - Namespace for the new format (oai_pod) that mixes both standard DC elements and any new ones
  - Namespace for the new elements (podterms)
- Namespaces are declared as URIs
- DCMI usage recommends use of Purl, but this is not required
- We will use:
  - http://yoowe.cms.hu-berlin.de/oaitutorial/oai_pod/
  - http://yoowe.cms.hu-berlin.de/oaitutorial/podterms/

Step 3: New Terms Schema

- Create an XML Schema for the new terms

  http://yoowe.cms.hu-berlin.de/oaitutorial/podterms/20040211/podterms.xsd

  (Notice the datestamp - makes it easier to enhance the schema without breaking things using the old one)
- Defines the new element ‘podterms:numberofpages’

Step 4: Container Schema

- Create an XML Schema for oai_pod record format

  http://yoowe.cms.hu-berlin.de/oaitutorial/oai_pod/20040211/oai_pod.xsd

  (Another date stamp!)
- Imports the dc Schema
- Imports the podterms Schema
- Defines a new container type oai_pod:elementContainer
  - dc elements (e.g. dc:identifier)
  - podterms element (podterms:numberofpages)
- Defines a container element ‘oaipod’ of type oai_pod:elementContainer

Step 5: Validate

- Create some test records
  (or modify your existing ones)

- Validate the records and schema with
  http://www.w3.org/2001/03/webdata/xsv/

Step 6: ListMetadataFormats

- OAI-PMH verb ListMetadataFormats
- Needs an awareness of the new format so:
- Need to modify your repository software (source code and/or configuration files) to support the new metadata format

```
<metadataFormat>
  <metadataPrefix>oai_pod</metadataPrefix>
  <schema>http://yoowe.cms.hu-berlin.de/oaitutorial/
   oai_pod/20040211/oai_pod.xsd
  </schema>
  <metadataNamespace>http://yoowe.cms.hu-berlin.de/
   oaitutorial/oai_pod/
  </metadataNamespace>
</metadataFormat>
```

Step 7: Other Verbs

- Also need to ensure oai_pod is available via requests
  - ListSets
  - ListIdentifiers
  - ListRecords
  - GetRecord
- Accept metadata prefix “oai_pod”
- Return the appropriate records

Step 8: Testing

- Use the Repository Explorer to test new format
- Ensure:
  - All requests work with the new ‘metadataPrefix’
  - oai_dc still works
  - appropriate records are returned
  - responses validate correctly

- Congratulations - you’ve got a new format!

Summary - Extending a format

- Decide a name and some namespaces
- Develop XML schema for the container and the new elements
- Create test records and validate
- Modify repository (source code and/or configuration files) to support new format
- Test and validate new repository output

oai_dc - isn't the MES I'm looking for

- Implement a different format e.g. IMS/IEEE LOM
- Very similar steps
- Already agreed names, XML schema and namespaces
- Should, therefore, be easier!

Implementing an existing format

- Modify the “ListMetadataFormats” response to include (e.g. for IMS):

```
<metadataFormat>
  <metadataPrefix>ims</metadataPrefix>
  <schema>http://www.imsglobal.org/xsd/
     imsmd_v1p2p2.xsd
  </schema>
  <metadataNamespace>
    http://www.imsglobal.org/xsd/imsmd_v1p2
  </metadataNamespace>
</metadataFormat>
```

- Extend other verbs to deal with ‘ims’ metadataPrefix

Summary

- OAI-PMH allows for any MES so long as...
  - ... it is encoded in XML with an XML Schema
- All repositories *must* support oai_dc for...
  - ...minimum level of interoperability
- If oai_dc is not enough - extend it!
- If oai_dc is not precise - wait a bit!
- If oai_dc is not ‘the one’ - use something else as well!

Tutorial
Open Archive Initiative

Conclusion

Links

- Open Archives Initiative
  http://www.openarchives.org
- OAI Metadata Harvesting Protocol
  http://www.openarchives.org/OAI/openarchivesprotocol.htm
- Virginia Tech DLRL OAI Project
  http://www.dlib.vt.edu/projects/OAI/
- Repository Explorer
  http://purl.org/net/oai_explorer
- NDLTD
  http://www.ndltd.org

More Links

- ARC Cross-Archive Search Service
  http://arc.cs.odu.edu/
- XML Schema Validator
  http://www.w3.org/2001/03/webdata/xsv
- Dublin Core Metadata Initiative
  http://www.dublincore.org
- E-Prints DL-in-a-box
  http://www.eprints.org
- XML Tools at W3C
  http://www.w3.org/XML/#software

Summary

During today's tutorial we hope that you have

- gained an overview of the history behind the OAI-PMH and an overview of its key features
- been given a deeper technical insight into how the protocol works
- learned something about some of the main implementation issues
- got an impression what to do in case oai_dc is not sufficient
- found some useful starting points and hints that will help you as implementers

Thanks

Question?

Contact:

Uwe Müller
Humboldt University Berlin
Computer and Media Service
+ 49 30 2093-7076
u.mueller@cms.hu-berlin.de