1. Introduction

- What is the OAI?
- Motivation
- General System Strategy
- History
- Case study: NDLTD

1.1. What is the OAI?

- What is the Open Archives Initiative (OAI)?
  - Organization dedicated to solving problems of digital library interoperability by defining simple protocols, most recently for the exchange of metadata.
- What is the Metadata Harvesting Protocol?
  - Protocol to transfer metadata from a source archive to a destination archive

1.2. Motivation

- Existence of some established but independent archives
- Need for cross-archive services (like search engines)
- Lack of low-cost interoperability technology
- Experience from past projects such as Dienst
1.3. General System Strategy

- Services
- Metadata Harvesting
- Document Model

1.4. History

- Santa Fe Meeting – October 1999
- Santa Fe Convention, January 2000
- Workshops (ACM-DL 2000, ECDL 2000)
- Structuring of the OAI
  - Steering Committee
  - Technical Committee
- Open Days – US/Europe
  - Metadata Harvesting Protocol, January 2001

1.5. Case Study: NDLTD

- Networked Digital Library of Theses and Dissertations
- Multiple independent university-based collections of electronic documents

- Virginia Tech
- Rhodes U.
- U. Waterloo
- OAI
  - Metadata Harvesting Protocol
- International ETD Library

---
2. Definitions / Concepts

- Basic Principles
  - What is an Open Archive?
  - Harvesting vs. Federation
  - Metadata vs. Data
  - Data and Service Providers

- Underlying Technology
  - HTTP and XML
  - XML, XML Namespaces and Schema

- Protocol Policies
  - Uniqueness and Persistence
  - What is a record?
  - Multiplicity of Metadata
  - Sets
  - Datestamp, Harvesting and Flow Control

2.1. 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
- ... aka OAI-Compliant Repository
- No implications for:
  - Physical storage of data
  - Cost of data
  - Metadata and data formats
  - Access control to server

2.2. Harvesting vs Federation

- Competing approaches to interoperability
  - Federation is when services are run remotely on remote data (e.g. Federated searching)
  - Harvesting is when data/metadata is transferred from the remote source to the destination where the services are located (e.g. Union catalogues)
  - Federation requires more effort at each remote source but is easier for the local system and vice versa for harvesting
  - OAI currently focuses on harvesting
2.3. 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

2.4. Data and Service Providers

- 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)

2.5. HTTP and XML

- Metadata Harvesting Protocol is an almost stateless request/response protocol
- Requests and responses are sent via the HTTP protocol
- Requests are encoded as GET/POST operations
- Responses are well-formed XML documents
2.6. XML Namespaces and Schema

- Consistency and data quality is ensured by using XML Schema descriptions for each possible response.
- XML Namespaces are used where necessary to clearly define which parts of the responses are actual metadata and which support the Metadata Harvesting Protocol.

2.7. Uniqueness and Persistence

- Each record must be uniquely addressable by a distinct identifier.
- Each metadata entity must be persistent to guarantee that service providers can always refer back to the source.

2.8. What is a record?

- A record refers to an independent XML structure that may be associated with digital or physical objects.
- Records are usually associated with metadata, not data.
- OAI advocates harvesting of records, which contain metadata and additional fields to support the harvesting operation.
2.9. Sample OAI Record

```xml
<record>
  <header>
    <identifier>oai:jcdl:tot3/identifier>
  </header>
  <metadata>
    <dc>
      <title>OAI Tutorial at JCDL</title>
      <creator>Hussein Soleman</creator>
      <language>English</language>
    </dc>
    <about>
      <metadataID>oai:jcdl:tot3/mdc/metadataID>
    </about>
  </metadata>
</record>
```

2.10. Multiplicity of Metadata

- Multiple formats of metadata allowed
- Dublin Core is mandatory
- Any other format allowed as long as it has an XML encoding
- E.g. MARC (Libraries), IMS (Education), ETDMS (Theses/Dissertations), RFC1807 (Bibliographies)

2.11. 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
- E.g. Subject areas, years, author names, search queries
2.12. Datestamps & Harvesting

Each record needs a datestamp that indicates its date of creation or modification.
Dates are used to allow for harvesting by date range, thus allowing incremental and continuous transfer of metadata from a data provider to a service provider.

2.13. Flow Control

HTTP “retry-after” mechanism can be leveraged to support server-side delaying of a client’s request.
Resumption Tokens can be used to return partial results – the client is issued with a token which may be presented to the server to receive more results.

3. Requirements to be a Data Provider

Source of metadata
Server technology
Datestamps
Deletions
Unique identifiers
Metadata mappings
3.1. Source of Metadata

- Database in proprietary format
- Collection of metadata records in well-defined format/s
  - Files on disk
- Metadata may be dynamically or statically extracted from data
- Synthetic collection

3.2. Server Technology

- WWW Server
- Protocol may be implemented in many forms
  - CGI Script (Perl, C++, Java)
  - Java Servlet
  - PHP
- Metadata (e.g., database) access mechanism required
- See www.openarchives.org for list of publicly available software templates
- See www.dlib.vt.edu for VT experimental software

3.3. 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
3.4. Unique Identifiers

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

3.5. Deletions

- Archives must keep track of deleted records, by identifier and datestamp
- All protocol result sets can indicate deleted records
- Deletions must be stored indefinitely so as to correctly propagate to service providers with varying harvesting schedules

3.6. Metadata Mappings

- Data provider must map its metadata to the formats it chooses to provide through its OAI interface
- Unqualified Dublin Core required
  - Best practice is to include a link to a human-readable page in the <identifier> tag
- Native formats recommended
- Community-based formats recommended
4. Metadata Harvesting Protocol

Service Requests
- Identify
- ListMetadataFormats
- ListSets
- GetRecord
- ListIdentifiers
- ListRecords

Metadata Multiplicity
- Date Ranges
- Resumption Tokens

4.1. Identify

Purpose
- Return general information about the archive and its policies

Parameters
- None

Sample URL
- http://www.anarchive.org/cgi-bin/OAI?verb=Identify

4.2. Identify - Response
4.3. ListMetadataFormats

**Purpose**
- List metadata formats supported by the archive as well as their schema locations and namespaces

**Parameters**
- identifier – for a specific record (O)

**Sample URL**

4.4. ListMetadataFormats - Response

4.5. ListSets

**Purpose**
- Provide a hierarchical listing of sets in which records may be organized

**Parameters**
- None

**Sample URL**
- http://www.anarchive.org/cgi-bin/OAI?verb=ListSets
4.6. ListSets – Response

Purpose
- Returns the metadata for a single identifier in the form of an OAI record

Parameters
- identifier – unique id for record (R)
- metadataPrefix – metadata format (R)

Sample URL

4.7. GetRecord

Purpose
- Returns the metadata for a single identifier in the form of an OAI record

Parameters
- identifier – unique id for record (R)
- metadataPrefix – metadata format (R)

Sample URL

4.8. GetRecord - Response

Purpose
- Returns the metadata for a single identifier in the form of an OAI record

Parameters
- identifier – unique id for record (R)
- metadataPrefix – metadata format (R)

Sample URL
4.9. ListIdentifiers

- **Purpose**
  - List all unique identifiers corresponding to records in the repository

- **Parameters**
  - from – start date (O)
  - until – end date (O)
  - set – set to harvest from (O)
  - resumptionToken – flow control mechanism (X)

- **Sample URL**
  - http://www.anarchive.org/cgi-bin/OAI?verb=ListIdentifiers&set=All

---

4.10. ListIdentifiers - Response

---

4.11. ListRecords

- **Purpose**
  - Retrieves metadata for multiple records

- **Parameters**
  - from – start date (O)
  - until – end date (O)
  - set – set to harvest from (O)
  - resumptionToken – flow control mechanism (X)
  - metadataPrefix – metadata format (R)

- **Sample URL**
  - http://www.anarchive.org/cgi-bin/OAI?
    verb=ListRecord&metadataPrefix=oai_dc&from=2001-01-01
4.12. ListRecords - Response

4.13. Metadata Multiplicity

4.14. Date Ranges
4.15. Resumption Token

5. Implementation Details

5.1. Tools Required
5.2. Basic program layout

parse WWW request to extract parameters
if (verb='Identify')
  ProcessIdentify;
else if (verb='ListMetadataFormats')
  ProcessListMetadataFormats;
else if (verb='ListSets')
  ProcessListSets;
else if (verb='GetRecord')
  ProcessGetRecord;
else if (verb='ListIdentifiers')
  ProcessListIdentifiers;
else if (verb='ListRecords')
  ProcessListRecords;
else
  Error (400, 'Unknown verb');

5.3. Object-Oriented Approaches

Cleaner separation of protocol, database access and metadata generation

Example approaches
- Each service request is handled by a object
  - Simpler incremental development
- Protocol, Database and Metadata are objects
  - Greater portability of code

5.4. Metadata Generation

Approaches
- Map from source to each metadata format
- Use crosswalks to generate additional formats

<table>
<thead>
<tr>
<th>source</th>
<th>dc</th>
<th>rfc1807</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>title</td>
<td>title</td>
</tr>
<tr>
<td>author</td>
<td>creator</td>
<td>author</td>
</tr>
</tbody>
</table>
5.5. Data Cleaning

- Escape special XML characters
- Convert to UTF-8 version of Unicode
- Convert entity references
- Remove extraneous whitespace
- Convert CR/LF for paragraphs
- URLs
  - #/?#=&;:+ must be encoded as escape sequences

5.6. Result Caching

- For multiple requests from many clients or to handle partial result sets
- Keep temporary tables/files
- Expire temporary data when no longer needed
- Is this necessary to handle date-range requests where new items are added to the result set while harvesting is in progress?

5.7. Error Handling

- 400
  - Illegal verb value
  - Illegal parameter values, combinations
- 404
  - Archive errors – cannot return a legal response
  - In general, everything else returns a legal but empty response!
5.8. Denial-of-Service Prevention

- Return only partial results and issue a resumption token for more
- Use 503 retry-after HTTP errors to have clients try again after a specified back-off time
- Use access control lists to limit who may access the archive
- Invoke an explicit delay before sending back results

5.9. Using Resumption Tokens

- Combine from/until/metadataPrefix/set and a record number indicator with delimiters into a sequential token
  For example:
  
  - from!until!metadataPrefix!set!recordnumber
  - 2000-01-01!2001-01-01!All!100
  - Use a session manager with automatic expiry
  For example:
  
  - vtetd123456789

6. Common Problems

- No unique identifiers!
- No dates stamps!
- Incomplete information in database
- New metadata format
- XML responses not validating
- Do I return an HTTP error or not?
6.1. 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
  *E.g. author+year+first word in title

6.2. 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

6.3. 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!
6.4. 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
- Register the new format with the OAI

6.5. XML not validating

- 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 character encoding issues

6.6. HTTP Error?

- Unless the archive is temporarily non-functional or the parameters are intrinsically wrong, do not return an HTTP error
  - If no metadata exists in a particular format, return a record with no metadata field
  - If the set name does not exist, return an empty list
  - If the identifier does not exist, return an empty response
7. Tools for Testing

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

7.1. RE Interactive Browsing

Open Archives Initiative - Repository Explorer

7.2. RE Parameter Testing
7.9. RE Multilingual Support

7.10. RE Automatic Test Suite

7.11. RE Error in Response
7.12. RE Error in XML

![XML Schema Instance Validation Error]

- Error in XML instance
- Schema error: "No schema instance found" [XML file: sample.xml]
- Example XML instance:

```
<book>
  <title>Sample Title</title>
  <author>John Doe</author>
  <year>2000</year>
</book>
```

7.13. OAI Registry

![The Open Archives Initiative]

The Open Archives Initiative
Registering as a Data Provider
- OAI-PMH Version 2.0
- Federation of Digital Libraries
- Data Provider Registration
- OAI-PMH Protocol
- Registration Service:
  - Registration Identifier: 123456
  - Registration Type: Repository
- Registration Details:
  - Repository Name: Sample Repository
  - URL: http://example.com/repo
- Registration Status: Active

7.14. OAI Registry

![List of Registered, OAI-Compliant Repositories]

- Repository List:
  - Repository 1
    - URL: http://example1.com
    - Type: Digital Repository
    - Registration Status: Active
  - Repository 2
    - URL: http://example2.com
    - Type: Digital Library
    - Registration Status: Active
- Registration Guidelines:
  - Must provide a clear metadata scheme
  - Must support at least one repository
  - Must be operational 24/7

7.15. XSV Schema Validator

Validator for XML Schema 20000922 version, XML Output

[Picture of XSV Schema Validator]

8. Service Providers

How to Harvest
- Policies
- Intermediate systems
- Tools
- Case Study: ARC
- Case Study: NDLTD

8.1. 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
8.2. Policies

- Use schedule for harvesting regularly
- Store date when last harvested (before you start)
- Use a two day overlap (or one day if you work with the timezone of the source)
  - New items may be added for the current day
  - Timezones create up to a day of lag if you ignore them
- Each time a record is encountered, erase previous instances

8.3. 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!

8.4. 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
8.5. Case Study: ARC

8.6. Case Study: NDLTD

Search/Browse Engines
- VTLS Virtua
- MARIAN
- Recommend
- Cross-Ref.

Other Services
- NDLTD ETD Union Catalog
- Virginia Tech
- Humboldt U.
- U. Oldenburg

9. OAI Communities
- Shared Metadata Formats
- Shared semantics
- Layering over OAI
- Closed OAI networks
- OAI within the DL
9.1. Shared Metadata Formats

- Use metadata formats accepted within a community to convey more specific information
- Examples
  - E-Print format (under development)
  - ETD-MS for theses and dissertations
  - VRA Core for multimedia
  - IMS Metadata for educational material

9.2. Shared Semantics

- Develop a shared understanding for the meanings of fields
- Examples
  - Developing controlled vocabularies for fields
  - Using specific fields for external links (OAI recommends using identifier in DC for this)
  - Choosing from among existing standards (like language names)

9.3. Layering over OAI

- Convert OAI records into more standard formats like MARC communications format
- Collapse multiple requests into one to make harvesting easier
- Name authority system (developed at OCLC) piggybacks name resolution over the OAI protocol
9.4. Closed OAI networks

- Data providers need not go public!
- Within an organization, OAI can be used for data transfer among heterogeneous systems
- More control over use, making global optimizations possible (like harvesting schedules and choice of metadata formats)

9.5. OAI within the DL

- Use the OAI protocol as the basis for components to communicate
- Examples
  - Search Engines could use dynamic sets to correspond to search results
  - Browsing can be directed by sets
  - Reviews and Annotations can each be independent OAI data providers

10. Now What?

- 1-2-3 Recipe
- Future of Metadata Harvesting Protocol
- Future of OAI
- Links
10.1. 1-2-3 Recipe

- DO I REALLY WANT TO DO THIS?
- Do I have an accessible metadata source?
- Do I have a server to host the OAI script/program?
- Can I satisfy the requirements to be a data provider?
- Can I write the code or modify a template or hire a programmer to do either?

10.2. Future of Protocol

- Version 1.1
  - Soon – minor upgrade to cater for updates to schema language by W3C
- Evaluation
  - Within a year – does this protocol make sense?

10.3. Future of OAI

- Advocacy for easier access to information
- New protocols/tools to support this mission
- Research projects to test theory underlying current architecture e.g. Cyclades
10.4. Links

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

10.5. More Links

- ARC Cross-Archive Search Service
  - http://arc.cs.odu.edu/
- XML Schema Validator
  - http://www.w3.org/2001/03/XMLSchema
- 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

That’s All Folks!