Joining the BRICKS Network - A Piece of Cake

Robert Hecht and Bernhard Haslhofer[1]

ARC Seibersdorf research - Research Studios
Studio Digital Memory Engineering
Thurngasse 8, A-1090 Wien, Austria
{robert.hecht|bernhard.haslhofer}@researchstudio.at

Abstract. Project BRICKS aims to build a Europe-wide distributed Digital Library in the field of Cultural Heritage. Each member organisation will run a node in the network (BRICKS node or BNode) and make its content available to the entire network. Content described in diverse Metadata standards can be searched with a unique query. Unlike other Digital Libraries, BRICKS is open and expandable, both on the organizational and the technical level. The aim of this paper is to demonstrate that the technical effort of joining the BRICKS network is minimal compared with the potential benefits of joining the BRICKS community.
This paper focuses on the BRICKS importer component, which is based on the Open Archives Protocol for Metadata Harvesting (OAI-PMH). After a short overview of the architecture of the importer and an outline of OAI-PMH, we list the technical requirements of the importer give a detailed list of the steps an institution take to join BRICKS. We also present first results obtained with our importer component.

1 Introduction

The EU project BRICKS aims to build a Europe-wide distributed Digital Library in the field of Cultural Heritage. The basis of this library will be a peer-to-peer network in which each member organisation runs one or several nodes in the network (BRICKS node or BNode) and makes its content available to the entire network [1]. The absence of central adminstration structures, although posing various implementation challenges, is seen as a major incentive for organisations to join the network: there will be no adminstration costs other than those for setting up and maintaining their own BNode. The software is available for free and will run on relatively inexpensive computers. Thus the cost of running a BNode will mainly be determined by the amount of work necessary to import and update content and metadata in the BNode. We are aware of the fact that this effort must be kept at a minimum to encourage institutions to join the network. In this paper, we describe the approach taken to achieve this goal.

The rest of this paper is structured as follows: in section 2 we give outline the architecture of the importer component and how it collaborates with other components of the system. In section 3 we present the Open Archive Initiative Protocol for Metadata

[1] An institution can still restrict access to its documents to certain users, but the technical infrastructure to access documents from every node is in place.

Harvesting (OAI PMH) and describe how we use it. Section 4 lists the steps an institution must carry out to import their metadata (and contents). Section 5 describes our experiences with the importer, and in section 6 we summarize our results.

2 Architecture

BRICKS consists of a number of loosely coupled components built on top of the peer-to-peer infrastructure. Each of these components is accessible via a web service interface. Software developpers can build applications using these components. The network aspects are hidden by the components - application developers don't need to worry about them.

The BRICKS Importer is an application which builds upon the following BRICKS components:

- **The Metadata Manager:** a repository for all *descriptive* metadata, i. e. metadata that describe content. Such metadata can include e.g. the title of a document, its author, the creation date, keywords, etc. The metadata descriptions are based on the Resource Description Framework (RDF) [1]. However, the Metadata Manager does not expose the RDF Graph, but summarizes the contained information into Records, which contain all data for a content item which conform to a given metadata schema.
- **The Content Manager:** the content repository of BRICKS, based on the Java Content Repository (JCR) [2]. The Content Manager can handle arbitrarily complex content models and store content that conforms to these models. The Content Manager offers two options to import external content:
  1. Importing the content: a copy of all content items is created and stored in the BRICKS Content Manager.
  2. Referencing content: the Content Manager stores a reference to the location of the content and retrieves it from the external system when a user wants to access it.

  It should be noted that a user who accesses a document will not notice if the Content Manager stores the actual content or just a reference to it.
- **The Collection Manager:** This component allows to organise content (and metadata) into a hierarchical system of sets - the so-called collections. The function of collections is more or less equivalent to that of folders / directories in a file system. Every BNode has at least one collection - the root collection. In principle, all content could be stored there, although this is strongly discouraged for organisational reasons (just like you wouldn't store all the files on your computer in the root directory).

The architecture of the Importer is outlined in fig. 2. The importer uses the OAI Protocol for Metadata Harvesting (PMH) to import metadata (see section 3). It acts as PMH client. The external system (the existing content management system) needs to implement or interface with an OAI PMH server (see section 4 for details). The retrieved metadata are stored in the Metadata Manager. The content (reference) is stored in the Content Manager.

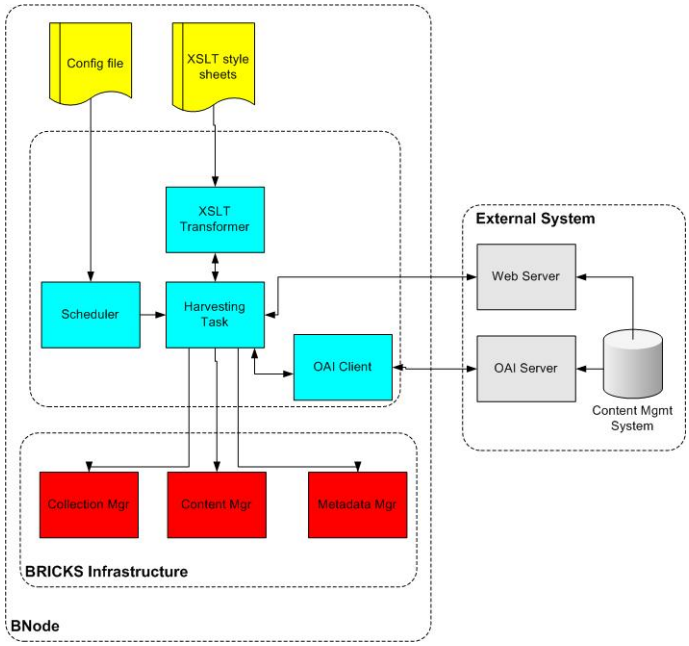

Fig. 1. Importer Architecture

The Importer is configured via a configuration file. In this file, the user needs to specify the address of the OAI PMH server, a schedule for harvesting, and the collection where the retrieved data should be stored. The specification of the target collection is very fine grained and can be made dependent on the metadata. E.g. it is possible to store all documents written by a certain author or all documents created in a certain period of time in one collection.

The scheduler generates Harvesting Tasks. Each Harvesting Task represents one pass of metadata harvesting with a certain set of parameters. The configuration file may define an arbitrary number of harvesting tasks. Harvesting tasks can be one-time (i.e at a defined date and time) or periodic, e.g. each Friday at midnight or on the second day of each month at 3:00 am.

Since the format of the OAI PMH is not RDF, but the Metadata Manager stores only RDF data, we need to translate the imported metadata to RDF before storing them. Mapping the XML serialisation of PMH to RDF/XML is rather straightforward and can be performed using XSLT [3] style sheets. This is performed by the XSLT Transformer component. For common standard metadata schemas, these style sheets will be provided as part of the BRICKS distribution. For other standards and proprietary schemas, they must be provided by the user. The workflow of an import is as follows:

1. The scheduler recognises that an import is due and creates a Harvesting Task.
2. The Harvesting Task triggers the OAI client to send a request to the OAI server.
3. The OAI client forwards the returned Metadata to the Harvesting Task.

4. The Harvesting Task creates content items corresponding to the returned metadata Records in the Content Manager. Depending on the settings in the configuration file, it either stores the location (e.g. URL) of the content item, or retrieves it from the web server of the external system and stores the content item themself. The content is put into a collection specified in the configuration file.
5. The Harvesting Task uses the XSLT Transformer to convert the metadata from the OAI PMH format to RDF/XML.
6. The Harvesting Task stores the converted metadata in the Metadata Manager.

3 OAI PMH

The Open Archive Initiative Protocol for Metadata Harvesting (OAI PMH) [4] is a protocol to exchange metadata that enjoys growing popularity in the Digital Library world. OAI PMH allows a Service Provider (client) to retrieve metadata from a Data Provider (server). It is implemented on top of HTTP and uses the internet as backbone.

OAI PMH structures metadata as Metadata Record. A Metadata Record contains all information concering a content item (e.g. a text document) that can be expressed with a specified standard (e.g. Dublin Core, VRA, MARC,...). If a content item is described in several standards, a separate record must be sent for each of them. Furthermore, OAI PMH requests that each content item has a unique identifier.

OAI PMH allows grouping information into (hierarchical) sets. E.g. a server for metadata of a museum might define sets like paintings, drawings, engravings, sculptures,... or medieval, renaissance, baroque,... Sets can overlap, i.e. a given content item can be an element of several sets. Sets can have subsets (arbitrary level of nesting supported), so a museum might divide their "'engravings"' set according to time periods, like egravings:baroque. Servers are not requested to support this feature, but if they do, it allows clients to selectively retrieve only the information they are interested in (e.g. a client could retrieve only items in the "'drawings"' set).

OAI PMH defines six Requests:

- *Identify:* Returns information about the server
- *ListMetadataFormats:* Returns a list of metadata formats supported by the server.
- *ListSets:* Returns the list of sets defined by the server (if any).
- *ListIdentifiers:* Retrieves the identifiers of items matching certain restrictions (e.g. on set, creation date, metadata format). If no restrictions are given, all identifiers are returned.
- *ListRecords:* Retrieves all Records of a given metadata format. Further restrictions, e.g. on set or creation date can be added.
- *GetRecord:* Retrieves a specific Record defined by its unique identifier and metadata format.

In the BRICKS Importer, we mainly use the ListRecords request.

The fact that OAI PMH allows to selectively retrieve records that were created, modified, or deleted in a given time-span enables "'incremental harvesting"', i.e. one can in a first pass harvest all metadata from a server and then run periodic updates in which only the changes are retrieved. Of course, this saves a lot of time.

OAI PMH Metadata Records can follow arbitrary metadata schemas, but they must include an unqualified Dublin Core description. Thus, organsiations that want to run an OAI server need to map their metadata to Dublin Core. However, this is typically not a problem, since Dublin Core is so generic that almost every existing Metadata Schema can be mapped to it. Moreover, it should be noted that this mapping is a one-time effort, since it is sufficient to define which fields in the existing descriptions should be mapped to which Dublin Core element.

4 Setting up the Import of external data

For institutions that want not only to use BRICKS but to contribute, the first step is to make their already existing external metadata and content available within the BRICKS network. We provide an Importer component which minimises the effort both for the first import of data from an external system to BRICKS and the synchronisation between the two systems. Rather than inventing a new communication protocol to achieve this, we have adopted the already accepted and well-established OAI PMH.

4.1 Requirements and first steps

In the import process, the external system acts as OAI server and BRICKS as OAI client. This means that the content providers must set up an OAI PMH server on top of their existing systems. Costs and effort to fulfil this requirement are minimal: on the one hand there exist several open source implementations (see [5] for a list of such systems and other interesting tools) that can be downloaded and installed for free, and on the other hand the OAI PMH is so simple that even non–experts can perform this task.

Since the OAI PMH specification foresees the unqualified Dublin Core [6] metadata format as a minimal requirement, and existing metadata are often stored in relational data bases, content providers only need to install an open source OAI server package and provide the mappings form their database tables to unqualified Dublin Core expressed as SQL statements. Since the metadata can change in the external system, we provide means not only for a one-time import, but also for continuous synchronisation. To reduce traffic between the server and client, the OAI server should support incremental harvesting, i.e. only records that were created, modified, or deleted since the last harvest are sent. To enable incremental harvesting, the OAI server should also support *persistent* information about deleted records.

As soon as the OAI-PMH server is set up at the content provider's side, the BRICKS Importer component can be configured and started to perform the (first) harvesting task. Each harvesting task must be specified in a *configuration record* and will be managed by a built-in scheduler. A configuration record holds the following information:

1. the URL pointing to the content provider's OAI Server
2. restrictions on the metadata records to be harvested
   - records pertaining to certain *Sets* (if supported by the OAI server)
   - records corresponding to a certain metadata format
   - records created, updated, or deleted in a certain time range
3. when to harvest;
   - month, day, and time (e.g. January 1st and July 1st at midnight), or
   - day and time (e.g. 3rd of every month at 3:00 pm), or
   - weekday and time (e.g. Monday and Friday at midnight), or
   - daily at a specified time (e.g. daily at 2:00 am), or
   - start harvesting immediately and do not repeat
4. to which BRICKS collection the harvested metadata records should be assigned to
   - *set-based*: specify a collection for all records pertaining to a certain set
   - *attribute-value-based*: all harvested metadata records matching a certain regular expression are stored in a specified collection
   - *default collection*: all metadata records that do not match any of the above criteria will be assigned to that collection
5. specify if external content should be imported as well, and where this content can be found. If the content is just a file, its location should be specified in the *dc:identifier* element, as recommended in the OAI PMH specification.
6. the path of stylesheet(s) which is used to transform harvested XML metadata records into RDF/XML
7. a unique identifier for the configuration record
8. what identifier to assign to the imported records

The format of the configuration file is XML, which is both popular and easy to understand and read. Typically, it should not be necessary to modify the configuration file, since the schedule for periodic updates can be defined when the file is first written. So even though it might take some effort to write the configuration file, this should be necessary just once.

Figure 2 shows a very basic configuration record which contains the necessary information to import all available unqualified Dublin Core metadata from one of the BRICKS consortium's content providers. It specifies the OAI server endpoint (`http://www.findsdatabase.org.uk/oai/service.php`), that the harvesting task should start immediately, and restricts the metadata to be harvested to unqualified Dublin Core (`http://www.openarchives.org/OAI/2.0/oai_dc/`). Further, it specifies that only the metadata but no contents should be imported and stored in a collection with the identifier *default*. Finally, the location of the stylesheet for transforming the harvested unqualified Dublin Core records into RDF/XML is given.

4.2 Advanced data import

Although importing unqualified Dublin Core metadata into the BRICKS network is a quick and straightforward, this is often not an optimal solution. This is because most

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ImporterConfiguration>
  <OAI_PMH_Request ID="BRICKS1">

    <ServerURL>http://www.findsdatabase.org.uk/oai/service.php</ServerURL>

    <Schedule>
      <Now/>
    </schedule>

    <Restrictions>
      <FormatRestriction>
        <Format
oaiPrefix="oai_dc">http://www.openarchives.org/OAI/2.0/oai_dc/</Format>
      </FormatRestriction>
    </Restrictions>

    <OAITargetPhysicalCollections>
      <DefaultCollection>default</DefaultCollection>
    </OAITargetPhysicalCollections>

    <ImportContents stopOnMissingDCIdentifier="false">false</ImportContents>

  </OAI_PMH_Request>

  <OAIFormatMapping>
    <Format
      oaiPrefix="oai_dc">http://www.openarchives.org/OAI/2.0/oai_dc/
    </Format>

    <MetadataXSLTPath>
        /org/bricks/impl/services/importer/misc/xslt/oai_dc2rdf_xml.xsl
    </MetadataXSLTPath>

    <ContentXSLTPath>
        /org/bricks/impl/services/importer/misc/xslt/oai_dc2oai_dc_jcrxml.xsl
    </ContentXSLTPath>
  </OAIFormatMapping>

</ImporterConfiguration>
```

Fig. 2. Sample configuration record for importing unqualified Dublin Core metadata

institutions are holding metadata that correspond to their proprietary schemas or some other standardized metadata schema, which are much more expressive than the Dublin Core schema. In most cases, mapping those schemas to unqualified Dublin Core causes an enormous loss of semantics in the metadata. As a result, the semantically weak metadata contributed to the BRICKS network do not provide the necessary basis for advanced, semantic–aware search and discovery mechanisms.

For that reason, we encourage institutions to contribute metadata corresponding to other standards than Dublin Core or to their own proprietary formats. Of course, this raises the required effort for importing external data, but from a future perspective it might be worth that effort. It should also be pointed out that most of the additional effort is one-time, because it involves the metadata schemas, not the individual metadata records.

The following steps are necessary if institutions want to contribute metadata that go beyond unqualified Dublin Core:

1. an XML Schema definition for the proprietary metadata schema (might be available for standards). This is useful for validating outgoing metadata records at the OAI Server side.
2. an OWL-DL ontology (Metadata Schema) matching the definitions in the XML schema. This ontology must be registered with the BRICKS Metadata Manager so that it can process the metadata.
3. an XSLT stylesheet for transforming OAI PMH metadata records into RDF/XML.
4. (optional) in case that a special or new JCR content model is desired, an XML schema is needed that reflects the content model.

5 Results

As a case study, we have imported several collections from two institutions that participate in the BRICKS project as content providers: the Austrian National Library's Pictures archive[2] and Consorzio Forma's image collection from their project "La Fortuna Visiva di Pompeii"[3]. Both institutions participating in the case study have existing content management systems in place and use metadata descriptions corresponding to proprietary schemas.

Since the OAI PMH has been designed with easy implementation in mind, the task of installing an already available OAI server implementation to handle OAI PMH requests took about one day of work by an experienced software engineer. We assumed that if the metadata in the existing systems are well organised, the task of accessing and deriving metadata should not be too onerous. This is especially true if an institution provides only unqualified Dublin Core Metadata. In our case study, our colleagues at Consorzio Forma have proved that this assumption indeed holds.

If institutions decide to provide semantically richer metadata than unqualified Dublin Core, the effort depends on the schema they use. If it is a standard (e.g. VRA [7], MARC [8], or CIDOC-CRM [9]) which has already been adopted in BRICKS, the effort remains nearly as low as when following the unqualified Dublin Core approach. However, if an institution employs a completely proprietary schema, the necessary XML schema, the OWL ontology as well as the stylesheet for transforming XML into RDF/XML must be created. In our case study, we were facing such a scenario when importing metadata from the Austrian National Library. However, it turned out that also this situation was manageable in collaboration between experts from boths sides.

6 Conclusions and Future Work

We have created an importer component for the BRICKS network which minimises both the cost (in terms of necessary hard- and software) and the amount of work needed

[2] `http://www.bildarchiv.at`
[3] `http://pompeii.sns.it`

to import existing data into the system. We have tested it with real-world size collections from two content providers and the results are quite satisfactory. Our future plans include to create BRICKS metadata schemas and XSLT style sheets for the most popular metadata standards in Cultural Heritage, so that most data can be imported with minimal effort. Moreover, we will provide a graphical user interface (GUI) for creating and updating the configuration file so that this task can be performed by persons who don't understand XML.

References

1. W3C: Resource Description Framework (RDF). (2005) http://www.w3c.org/RDF.
2. java community process: JSR 170: Content Repository for Java technology API. (2005) http://www.jcp.org/en/jsr/detail?id=170.
3. W3C: XSL Transformations (XSLT) Version 1.0. (2005) http://www.w3.org/TR/xslt.
4. Open Archives Initiative: The Open Archives Initiative Protocol for Metadata Harvesting. (2004) http://www.openarchives.org/OAI/openarchivesprotocol.html.
5. Open Archives Initiative: OAI tools. (2005) http://www.openarchives.org/tools/tools.html.
6. The Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1: Reference Description. (2004) http://dublincore.org/documents/dces/.
7. Visual Resources Association Data Standards Committee: VRA Core Categories, Version 3.0. (2002) http://www.vraweb.org/vracore3.htm.
8. Library of Congress - Network Developement and MARC Standards Office: MARC Metadata Standard. (2005) http://www.loc.gov/marc/.
9. CIDOC Documentation Standards Working Group: CIDOC Conceptual Reference Model (CRM). (2005) http://cidoc.ics.forth.gr/.