DELIVERABLE

Project Acronym: EuDML
Grant Agreement number: 250503
Project Title: The European Digital Mathematics Library

Deliverable 5.4 – The EuDML Metadata Registry and Repository - Final

Revision: 1.0

Authors:
Gilberto Pedrosa IST
José Borbinha IST
João Edmundo IST
António Higgs IST

Contributors:
Romeo Anghelache FIZ
Nicolas Houillon UJF/CMD
José Monteiro IST
Claudia Antunes IST
André Vasconcelos IST
João Lopes IST
Gabriel Pestana IST
Alberto Silva IST

Project co-funded by the European Commission within the ICT Policy Support Programme

<table>
<thead>
<tr>
<th>Dissemination Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

Sunday, 10 February 2013
Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organisation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>02-01-13</td>
<td>Gilberto Pedrosa</td>
<td>IST</td>
<td>Initial document structure</td>
</tr>
<tr>
<td>0.2</td>
<td>21-01-13</td>
<td>António Higgs</td>
<td>IST</td>
<td>Content addition</td>
</tr>
<tr>
<td>0.3</td>
<td>22-01-13</td>
<td>João Edmundo</td>
<td>IST</td>
<td>Add of REPOX Content</td>
</tr>
<tr>
<td>0.4</td>
<td>28-01-13</td>
<td>Romeo Anghelache</td>
<td>FIZ</td>
<td>Internal review</td>
</tr>
<tr>
<td>0.5</td>
<td>29-01-13</td>
<td>Nicolas Houillon</td>
<td>UJF/CMD</td>
<td>Internal review</td>
</tr>
<tr>
<td>0.6</td>
<td>08-02-13</td>
<td>Alan Sexton</td>
<td>UB</td>
<td>Internal review</td>
</tr>
<tr>
<td>1.0</td>
<td>10-02-13</td>
<td>José Borbinha</td>
<td>IST</td>
<td>Final version</td>
</tr>
</tbody>
</table>

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.
## Document Index

1. Introduction .............................................................................................................................................. 1
2. Metadata Repository ................................................................................................................................... 2
   2.1. Introduction ........................................................................................................................................ 2
   2.2. REPOX as a repository ......................................................................................................................... 2
   2.3. Architecture of Services ....................................................................................................................... 3
   2.4. Information Model ............................................................................................................................... 4
   2.5. Services and Interfaces ......................................................................................................................... 6
   2.6. External Services ................................................................................................................................ 7
   2.7. Technology Used .................................................................................................................................. 9
3. Metadata Registry ....................................................................................................................................... 10
   3.1. Introduction .......................................................................................................................................... 10
   3.2. Primary Purposes ................................................................................................................................. 10
   3.3. Use Cases ............................................................................................................................................ 10
      3.3.1. Use Case Description .................................................................................................................... 10
      3.3.2. Alignment with the ISO 11179 ....................................................................................................... 13
   3.4. Graphic User Interface ....................................................................................................................... 13
      3.4.1. Interface Overview ...................................................................................................................... 13
      3.4.2. Item Discovery .............................................................................................................................. 14
      3.4.3. Item Administration ..................................................................................................................... 15
      3.4.4. Schema Versioning ...................................................................................................................... 16
      3.4.5. Overview of the XMApper’s Interface ......................................................................................... 16
4. Full-Text harvesting service ..................................................................................................................... 17
5. Results .......................................................................................................................................................... 19
6. Conclusions ................................................................................................................................................ 20
7. References ................................................................................................................................................... 21
1. Introduction

This document is an update of the Deliverable D5.1 which describes the REPOX framework in greater depth. It follows the original text whenever appropriate, but just a small portion of the content is overlapped in order to create some context.

This document describes the overall architecture and the key components of both the REPOX and MDR frameworks. Together with its complementary appendix, it is intended as the general reference document for development purposes.

This is a concise document focused on explaining not only the purpose of REPOX as the EuDML metadata repository and MDR as its metadata registry, but also how their integration with external services like Full-Text, Record Validator and EuDML Transform can fit together to create a seamless harvesting process from the raw data made available by the content providers until it is stored within the EuDML repository.
2. Metadata Repository

2.1. Introduction

REPOX is the technology supporting the data aggregator service, as part of the EuDML Repository. REPOX manages the harvesting processes of the metadata and full-text provided by the Data Providers, plus the processes for data transformation and normalization [2].

In brief, REPOX provides the following main functionalities:

- The registration of data providers, their collection descriptions (one data provider might make available to EuDML more than one collection), and the configurations for the harvesting of the relating metadata and full-text.
- The automatic and manual harvesting of the metadata by OAI-PMH (according to configurations and options provided by the data providers and the decisions of the administrators of the central service) or by HTTP or FTP. Support is provided for multiple metadata formats. The harvesting of the full-text is expected to be carried out via HTTP or FTP.
- Monitoring of the quality of service of the OAI-PMH servers, including statistics.

2.2. REPOX as a repository

The REPOX is the metadata repository for EuDML. It is a core component of the EuDML metadata ingestion workflow from the Content Providers to the YADDA Storage System (EuDML main data repository) (Figure 1).

![Figure 1: Workflow of the metadata ingestion process to EuDML](image)

The ingestion process is initiated with the configuration of each data provider and data set for its correspondent content provider, and concluded with the ingestion of the data (The whole procedure can be done both through the graphical web interface or the web REST based API [2]). However, new content providers must contact the REPOX administrator so that their collections are registered in REPOX.
Then, REPOX verifies if the data needs a transformation. If so, REPOX orders the EuDML Transform service to harvest it from the content providers and then stores the results in its databases. Otherwise, if no transformation is required, the data is ingested directly from the content providers into REPOX.

After the data is configured and ingested, it is transformed (if not done so during the ingestion as described previously) into the EuDML public NLM (Two schemas are used to expose the original records received from the content providers in a uniform way) with the help of the EuDML Transform service. The result is then validated by the Record Validator service against the EuDML public schema and, if any errors occur, they are reported to the content providers and the process stops. However, if the content is valid, the data is transformed into the EuDML internal NLM (a schema for internal use inside the EuDML harvesting system) and once again validated against the EuDML internal schema. If every operation is successful thus far, the full-text content is then retrieved from the internal data. At last, all the metadata and full-text content is stored within the YADDA Storage System.

### 2.3. Architecture of Services

![Figure 2: REPOX Services](image)

The REPOX basic services (Figure 2) are:

- The **Data Manager** service aggregates the core components of the system:
- The **Data Provider Manager**
- The **Data Set Manager**
- The **Harvester Manager** is the service responsible for harvesting the records from the data providers. It can use one of the following interfaces: OAI-PMH client, HTTP-get, file system folder or FTP client. REPOX can harvest files in ISO2709, MarcXchange,

1 [http://eudml.org/schema/2.0/eudml-article-2.0.xsd](http://eudml.org/schema/2.0/eudml-article-2.0.xsd)  
ESE, NLM or ultimately any XML format. For each record harvested, the Data Manager component creates and stores the indexes of the access points in Data Set Repository.

- The **Data Providers Repository** consists of a set of XML files registering the necessary information about the Data Providers, their Data Sets, and the respective ingestss.
- The **Data Set Repository** is a database storing the records of the data sets ingested.
- The **Full-Text Manager** is the service responsible for analysing the ingested records and harvesting the contents linked from them, which is stored in the **Full-Text Repository**.
- The **Full-Text Repository** provides the local storage for the harvest full-test. In its simplest version it is just a local system, but if required it can be replaced by a specific alternative repository.
- The **Data Manager Exporter** makes bibliographic metadata available to external systems through the interface OAI-PMH server. It also allows external access to those records through the file system, as also the access to the full-text. An example of an external client of this service is the **Profiling Service**, which can provide basic profiling for each data set of interest.
- The **Content Transformation Manager** manages the transformations between metadata formats (schemas) and is also responsible for the application of those transformations to specific metadata sets. Furthermore, it is complemented with an MDR - Metadata Registry, a service to store, support management of and provide XSLT transformations of datasets, namely to eudml-article2.
- The **Repox2Yadda** is a service that communicates with YADDA Storage using the corresponding API. The metadata and the full-text harvested by REPOX are stored in YADDA Storage under the control of this service. This service is included according to the specific requirements of the EuDML project.
- The **Yadda2Repox** is a service that communicates with YADDA Storage using the corresponding API. The metadata consolidated by EuDML and stored in YADDA Storage that is to be made available to external partners (typically by OAI-PMH) is ingested by this service.

### 2.4. Information Model

Figure 3 shows the REPOX Information Model (which is shared by the services).

Data Providers are entities with one or more collection of records (record sets), each with a Data Source.

Data Sources are either OAI-PMH or Directory Importer, the first meaning that the records will be harvested from an OAI-PMH server and the second meaning a folder in the file system. To ingest from folders, REPOX recognizes three strategies: Simple File Extract, ISO2709 File Extract and MarcXchange File Extract. Simple File Extract is the default method, where there is no processing of the XML records. The only associated logic is the validation of the XML file itself. ISO2709 File Extract and MarcXchange specifically target those formats. ISO2709 File Extract requires the file Character Set and the format variant because even though ISO2709 is a standard, some institutions do not follow it exactly. Because ISO2709 is not an XML format and REPOX only handles XML, the format is ingested as MarcXchange because there is no data loss in that transformation. In the three scenarios the files may be zipped and unzipped prior to ingestion.
Data Sources can have associated ScheduledTasks, by scheduling an Ingest of records or an export of the records to the file system. A Task is a managed action in REPOX. Scheduled Tasks are tasks that occur at specific times with a periodicity (unique harvest, daily, weekly and every number of months).

Access Points (AP) enable the retrieval of the records by more than only their identifiers. For that purpose, Access Points are associated to Data Sources, to define how to process the pertinent information for indexing. These AP are used by the AccessPointsManager (APM) to extract the relevant data from each record and build the respective indexes. Those indexes are maintained in a relational database for efficiency, as they are not part of the fundamental model.

A Metadata Transformation is a translation between two metadata formats (ex: local schema to NLM). Every Data Source can have any number of transformations. The transformations are stored as XSLT files, even though it is possible to create a visual mapping of them which is stored in an intermediate XML format internally to allow editing. The records can be retrieved by OAI-PMH in their original format or any format which has been configured a mapping to. The mapping is performed by request and not stored, because the performance impact is not noticeable.

The record identifiers used in REPOX can be associated in two ways: generated by REPOX or extracted from each record using an XPath expression. The advantage of using extracted identifiers is that it is possible to update just the changes because the records can be recognized by the identifier.
2.5. Services and Interfaces

The REPOX has a graphic user interface (example in Figure 4) to be used by human beings. REPOX also provides to the outside a set of REST services for Data Providers (Table 1), Data Sources (Table 2), Records (Table 3) and Statistics (Table 4).

**Figure 4: REPOX graphic user interface**

- Retrieve all available Data Providers
  - Syntax: "REPOX installation"/rest/dataProviders/list
- Creates a new Data Provider
  - Syntax: "REPOX installation"/rest/dataProviders/create
- Updates a Data Provider
  - Syntax: "REPOX installation"/rest/dataProviders/update
- Deletes a Data Provider
  - Syntax: "REPOX installation"/rest/dataProviders/delete

Table 1: Data Provider Operations

---

The services detailed in these tables provide a web API based on a REST architecture to enable machines to interact with the REPOX framework.
Retrieve all available Data Sources
  - Syntax: “REPOX installation”/rest/dataSources/list

Creates/Updates a Data Source from OAI-PMH server
  - Syntax: “REPOX installation”/rest/dataSources/{createOai/updateOai}

Creates/Updates a Data Source from FTP server
  - Syntax: “REPOX installation”/rest/dataSources/{createDataSourceFtp/updateDataSourceFtp}

Creates/Updates a Data Source from HTTP server
  - Syntax: “REPOX installation”/rest/dataSources/{createHttp/updateHttp}

Creates/Updates a Data Source from local folder
  - Syntax: “REPOX installation”/rest/dataSources/{createFolder/updateFolder}

Deletes a Data Source
  - Syntax: “REPOX installation”/rest/dataSources/delete

Starts the Data Source ingestion
  - Syntax: “REPOX installation”/rest/dataSources/startIngest

Stops the Data Source ingestion
  - Syntax: “REPOX installation”/rest/dataSources/stopIngest

Schedules a Data Source ingestion
  - Syntax: “REPOX installation”/rest/dataSources/scheduleIngest

Retrieves the list of all schedules from a specific Data Source
  - Syntax: “REPOX installation”/rest/dataSources/scheduleList

Retrieves the list of all harvesting Data Sources
  - Syntax: “REPOX installation”/rest/dataSources/harvesting

Retrieves the Data Source status
  - Syntax: “REPOX installation”/rest/dataSources/harvestStatus

Retrieves the last Data Source ingest log
  - Syntax: “REPOX installation”/rest/dataSources/log

Table 2: Data Source operations

Retrieves a specific record
  - Syntax: “REPOX installation”/rest/records/getRecord

Save record
  - Syntax: “REPOX installation”/rest/records/saveRecord

Delete record (mark as deleted)
  - Syntax: “REPOX installation”/rest/records/deleteRecord

Erase Record (remove permanent)
  - Syntax: “REPOX installation”/rest/records/eraseRecord

Table 3: Record operations

Retrieves statistics for data sets of type INTERNAL/EXTERNAL/YADDA or ALL

Table 4: Statistics operations

2.6. External Services

The REPOX has a protocol based on the REST architecture so that it can use external web services. All web services that follow this protocol can be integrated within the harvesting process for each data set. External Services can be easily added/edited/removed using the REPOX External Services Manager (Figure 5).
Each External Service contains an identifier, a URI which is used to call the service and, when it takes a long time to execute, a status URI which is used to monitor its progress (Figure 6). It also contains a set of parameters which can be passed through a “text box, boolean or combo field” (Figure 6). These specifications define an external service that can then be used during the harvesting process of any data set.

After the definition of the External Service it can be added to the data set (Figure 7) in which each service parameter is filled with the real input value. At this stage the user might also choose to run/not run the external service during the harvesting process by changing the “Enabled” option.
In this project we used this mechanism for three web services:

- **Record Validator** – A service that, before the harvest is performed, evaluates the record’s metadata according to a set of schemas and provides information to the data providers of the missing information so that it can be fixed.
- **EuDML Import** – A service that applies a set of custom enrichments and transformations to the records before they are harvested.
- **Full-Text Harvester** – A service that, after the records are harvested into REPOX, goes through each record analysing it and harvesting the contents linked from them (PDFs mostly).

This External Services Architecture gives the REPOX the flexibility needed to adjust to custom changes that may arise in the future.

### 2.7. Technology Used

- **JAVA**
- **PostgreSQL or Apache Derby** – used as the relational database of the Data Set Repository.
- **Tomcat or Jetty server** – used for operational running.
- **Google Web Toolkit** – Java web development framework for user interfaces.
- **Xalan-Java** – an XSLT processor for transforming XML documents.
- **Apache Maven** – software project management and comprehension tool.
3. Metadata Registry

3.1. Introduction

The REPOX Metadata Registry (MDR) is a REPOX’s service designed to store and maintain in a controlled environment the range of data models used within an organization, and how these models relate to others. As a result, the REPOX MDR promotes a common understanding of the information managed within an organization and assists organizations in the sharing and exchanging of mutually agreed information. The REPOX MDR allows the viewing and editing, to some extent, of the stored transformations through its integrated model mapper (the XMApper).

The MDR stores data schemas and transformations which are referred to as Administered Items.

3.2. Primary Purposes

The primary goals of the REPOX MDR are to improve metadata interoperability by allowing:

- The documentation of the data models from providing organizations and the relationships among their attributes;
- Transformations among the various data models.

3.3. Use Cases

3.3.1. Use Case Description

The REPOX MDR currently has 1 class of actors.

The Registered User is able to use all of the system functions: discovery, access and maintenance of the Administered items. Maintenance involves editing, deleting and the responsibility to identify, describe and register new Administered Items following the registration requirements. A registered user can also view, create and edit transformations created with the XMApper model mapping module.

The following diagram defines the main use cases of the REPOX MDR system.

Figure 8: Overview of the use cases defined for the REPOX MDR
## Use Case UC1 – Discover Administered Items

**Description**
A Registered User searches or browses the REPOX MDR for a particular Administered Model. For data schemas, searching is by schema name or namespace, or a part thereof. For transformations, searching can be by transformation ID, source or destination format name, or a part thereof. Results for all searches are shown on-the-fly (constantly refreshing while the user is typing) near the searching box. Browsing is by page, with the maximum number of elements presented on each being defined by the user.

<table>
<thead>
<tr>
<th>Primary Actor</th>
<th>Registered User</th>
</tr>
</thead>
</table>

### Basic Flow
1. The user types a searching term in the searching box.
2. Select one of the displayed results.
3. The view is updated and shows only the intended result.

## Use Case UC2 - Access Administered Items

**Description**
When accessing a data schema the user gets its name, namespace, link, and the data sets and transformations that use that schema (with this information being present for each of the schema’s versions). When accessing a transformation the user can see its ID, description, source and destination formats, see which data sets use the transformation and download the transformation script file. The user can also check if the transformation is fully compliant with the MDR\(^3\) and if it’s editable in the XMApper.

<table>
<thead>
<tr>
<th>Primary Actor</th>
<th>Registered User</th>
</tr>
</thead>
</table>

### Basic Flow
1. The user discovers the desired Administered Item.
2. The user accesses the Administered Item

## Use Case UC3 – Administer Items

**Description**
A Registered User can register new and retire or change existing Administered Items. Successfully editing and saving a transformation will make it MDR compliant.

<table>
<thead>
<tr>
<th>Primary Actor</th>
<th>Registered User</th>
</tr>
</thead>
</table>

### Basic Flow
**Register:**
1. Request creation of a new Administered Item.
2. Enter the necessary information.
3. Save the new item.

**Retire:**
1. Select the desired Administered Item.
2. Request the deletion of the Administered Item.

\(^3\)The REPOX MDR accepts transformation using the old REPOX definition for transformations, allowing them to be used in datasets. However a warning is placed upon them, meaning that a Registered User has to review and choose their source and destination formats (from the MDR database) for them to become fully compatible with the REPOX MDR.
<table>
<thead>
<tr>
<th>Use Case</th>
<th>UC4 – Manage Schema Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A Registered User can add new and edit or delete existing versions of a data Schema registered in the MDR. At least one version must exist at all times. Versions currently in use (in data sets or transformations) can’t be deleted.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Registered User</td>
</tr>
<tr>
<td>Preconditions</td>
<td>A data schema must have been selected.</td>
</tr>
<tr>
<td>Basic Flow</td>
<td>1. The user edits the schema properties.</td>
</tr>
<tr>
<td></td>
<td>1. Select “Add new version” and enter the necessary information.</td>
</tr>
<tr>
<td></td>
<td>2. Edit existing version(s) properties.</td>
</tr>
<tr>
<td></td>
<td>3. Delete desired version(s).</td>
</tr>
<tr>
<td></td>
<td>2. Save changes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case</th>
<th>UC5 – View Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A Registered User can graphically view a transformation between two schemas through the use of the MDR’s model mapper (the XMApper). As for now, only transformations created with the XMApper can be viewed.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Registered User</td>
</tr>
<tr>
<td>Preconditions</td>
<td>The transformation must have been created with the XMApper</td>
</tr>
<tr>
<td>Basic Flow</td>
<td>1. The user opens the desired transformation with the XMApper</td>
</tr>
<tr>
<td></td>
<td>2. The user views a graphic representation of that transformation and its related schemas.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case</th>
<th>UC6 – Manage Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A Registered User can create and define new transformations with the XMApper, as well as edit previously mapper created transformations.</td>
</tr>
<tr>
<td>Primary Actor</td>
<td>Registered User</td>
</tr>
<tr>
<td>Basic Flow</td>
<td>Create:</td>
</tr>
<tr>
<td></td>
<td>1. The user chooses the source and destination schemas.</td>
</tr>
<tr>
<td></td>
<td>2. Creates the mappings between the two schemas.</td>
</tr>
<tr>
<td></td>
<td>3. Saves the new transformation by entering the necessary information.</td>
</tr>
<tr>
<td>Edit:</td>
<td>1. The user selects the desired transformation.</td>
</tr>
<tr>
<td></td>
<td>2. Edits the mappings of that transformation.</td>
</tr>
</tbody>
</table>
3. Saves the changes overwriting the old version.

3.3.2. Alignment with the ISO 11179

The ISO/IEC JTC1 SC32 WG2 develops standards for metadata and related technology\(^4\), namely the ISO/IEC 11179, the standard that defines the concepts behind a Metadata Repository. Although being substantially different, the REPOX MDR implementation has its item administration bases established in the suggestions of this standard:

- **Items Administration** – All the base management mechanisms defined by the ISO/IEC 11179 (see [1]) are available for the Administered Items of the REPOX MDR (data schemas and transformations): Register, Retire and Change. They are mostly similar, with the notable exception of the Retire mechanism, in which the legacy items (predicted in the standard) are not implemented in the REPOX MDR, and retiring an item will delete it from the system.

- **Schema Versioning** – The ISO/IEC 11179 indicated that each Administered Item may require a different versioning treatment. In the REPOX MDR a dedicated versioning system is available for data schema versioning.

3.4. Graphic User Interface

3.4.1. Interface Overview

The REPOX MDR interface, and the access to its functionalities, is as follows.

![Figure 9: The REPOX MDR Schemas section interface](image)

Figure 9 identifies the main components of the general REPOX MDR interface, and the specific functions of the data schemas section.

REPOX MDR general interface:

- **2** – The button in the main toolbar of the REPOX, which grants access to the MDR (cannot be seen by anonymous users).
- **5** – The button group, which allows the user to switch between the MDR’s Schemas (data schemas), Mappings (transformations) and the XMApper sections.

Functions of the data schema section:

\(^4\)http://metadata-standards.org/
1 – Schema management functions (Register, Change and Retire).
3 – Schema browsing toolbar.
4 – Schema searching box.

Similarly, Figure 10 shows all the same functions, but now from the transformations section perspective. Notice now how the selected section in the MDR is “Mappings” (#1).

3.4.2. Item Discovery

The REPOX MDR has two main mechanisms for discovering Administrated Items: searching and browsing.

The searching function is available through the search bar, which is always available when viewing schemas or transformations. The user enters a searching term and the results are automatically displayed below the searching bar. The user can then select one of the results, which will immediately update the view to show only that item. The clear button clears the searching filter, restoring the original all-items view.

Simple browsing of the Administered Items by page is also available, and controlled by simple commands:

1. Search bar
2. Clear search button
3. Search results
4. Access first page
5. Access previous page
6. Enter custom page (press «Enter» to apply)
7. Access next page
8. Access last page
9. Refresh current page
3.4.3. Item Administration

Schemas and transformations have their own menus for creation and editing (Figure 12 and Figure 13).

![Figure 12: Schema details menu](image)

The schema’s menu is divided in two groups, first the Schema Information group, with general information to be entered by the user, and second the Schema Versions group, which will be described in the next section of this document. Fields marked with a red asterisk are required to be able to create the schema/save the changes.

![Figure 13: Transformation details menu](image)

In the transformation’s menu the user can enter all the information regarding the transformation, including the selection of source and target formats from the MDR’s database (#1) and
uploading the transformation file from his file system (#3). More details about the selected source and destination formats can be seen in the info-boxes below the fields (#2). As with schema’s menu, the fields marked with a red asterisk are required for being able to save.
Deleting both schemas and transformations can be achieved via the general interface, shown in Figure 4 and Figure 6 (buttons “Delete” for schemas; and “Remove Transformation”), with some restrictions applying: schemas currently in use by transformations or datasets cannot be removed; Transformations currently in use by datasets can be removed, but the user is warned that the transformation will also be removed from the dataset.

3.4.4. Schema Versioning

As for schema versioning, when editing the properties form of a schema the user has access to its versioning system (Figure 14).

The schema versioning operates through simple commands:

1. Delete version (a confirmation prompt will be displayed). When it’s grayed out it means that the version is currently in use. When only version is available the button will not appear (at least one version must always be available).
2. Add a new schema version.
3. Save or cancel changes (all the form requirements, marked with a red asterisk, must be met to be able to save).

3.4.5. Overview of the XMApper’s Interface

The REPOX MDR comes with a mapping module, called XMApper, enabling the user to view and edit transformations. NOTE: Only transformations previously created with the XMApper can be opened.
Figure 15: The XMApper main interface

Figure 15 shows some of the main components of the XMApper interface:

1. Representation of the source schema.
2. The schemas’ toolbar with functions for viewing and navigation.
3. The mapping board where visual connections can be made between the schemas.
4. Button to access to the preferences.
5. Button to save the transformation.
6. Representation of the destination schema
7. Properties of the selected mapping board element.
8. Button to access the XMApper and its menu options.

4. Full-Text harvesting service

An important part of the EuDML project was to collect all the OCR made available by partners and then retrieve its mathematical formulas and text from it, with the purpose of indexing them and thus improving overall search services.

In order to automate the processes, the content providers were expected to provide links in the metadata records to their full-text material.

The Full-Text Harvester is the service responsible for analyzing the ingested records and harvesting the contents linked from them (mostly PDFs in this project). It is composed by a web API which exposes all its features through a REST based protocol, and a graphical user interface so that users can visually manage and monitor its harvests (Figure 16).
The harvester saves all the collected data into YADDA storage system. Furthermore, all the information about the harvest process can be seen in the Full-Text Main Screen (Figure 16). Also, new harvests can be created through the OAI-PMH protocol or using a local folder in the file system (Figure 17).

When using the OAI-PMH protocol (Figure 18), the metadata format can retrieved from the OAI server specified in the OAI-URL field, or chosen amongst a set of pre-defined formats managed within the Full-Text Harvester through the Metadata Format Manager. Each of these metadata formats can have pre-defined XPath values (in EuDML project were defined “eudml-article2” and “eudml-book2” formats) which will be suggested to the user during the creation of a new harvest.
Figure 18: Metadata Format Manager

Some additional features of the Full-Text Harvester are:

- **Error recovery** – It creates a list of failed records so, when it harvests again, only the failed records are harvested instead of the whole collection.
- **HTML interpretation** – HTML documents are parsed so that PDF links can be extracted.
- **METS interpretation** – If the links inside the records are references to METS files, the Full-Text Harvester recognizes the METS format and can retrieve the PDF files inside it.
- **Harvest Logging** – Each harvest has a log that contains detailed information about the process, including: start and end times; duration; number of records; status; file number; total harvested information size; failed records;
- **INFTY Results Harvest [3]** – Besides the PDFs, the Full-Text Harvester also harvests, for each record, its corresponding INFTY resultant files in the MathML, Latex and KML formats.

**5. Results**

During the course of the project many records, full-text and INFTY content files were collected for each of the EuDML project’s collections (Table 5).

<table>
<thead>
<tr>
<th>Collection</th>
<th># Records</th>
<th># Full Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>BulDML</td>
<td>715</td>
<td>715</td>
</tr>
<tr>
<td>CEDRAM</td>
<td>2.084</td>
<td>2.084</td>
</tr>
<tr>
<td>DMLE</td>
<td>6.401</td>
<td>6.296</td>
</tr>
<tr>
<td>DML_CZ_Proceeding</td>
<td>1.458</td>
<td>2.916</td>
</tr>
<tr>
<td>DML_CZ_Monograph</td>
<td>108</td>
<td>3.719</td>
</tr>
<tr>
<td>DML_CZ_Serial</td>
<td>28.662</td>
<td>24.443</td>
</tr>
<tr>
<td>ELibM</td>
<td>36.835</td>
<td>32.814</td>
</tr>
<tr>
<td>GALLICA</td>
<td>2.081</td>
<td>2.081</td>
</tr>
<tr>
<td>NUMDAM</td>
<td>50.240</td>
<td>50.240</td>
</tr>
<tr>
<td>NUMDAM_book</td>
<td>426</td>
<td>426</td>
</tr>
</tbody>
</table>
The table above shows that all the metadata was collected successfully. On the other hand, some collections like DMLE, ElibM, etc. had some records from which no full-text content was retrieved because it wasn’t available. Also, some providers like EDP didn’t allow their full-text content to become public, restricting its retrieval and INFTY processing.

Other reasons for not having results include:

- The quality of the full-text content is too low to be processed by INFTY;
- Given the time INFTY takes to process each content and the size of some large collections, more time was required than expected, which means that some collections GDZ are still running.

### 6. Conclusions

Based on the combined effort of all project partners, a seamless harvesting process was developed that collects and transforms/enriches the metadata and full-text content made available by the content providers. This harvesting process is able not only to continuously store and update this content, but also to give feedback to the content owners if something is missing or wrong.

The REPOX framework is used as the core metadata harvester of the EuDML project and was easily able to cope with data enrichment, full-text harvesting and validation requirements through its External Services protocol based on a REST architecture.

The Metadata Registry integrated into REPOX allowed an additional layer of management of the schemas and mapping used by data sets within REPOX. Therefore, it enables management of schemas and mappings, and includes the creation of new mappings between schemas already existing schema transformation files or by visually defining a new one through a web graphical interface.

The Full-Text Harvester was developed and used as an external service for REPOX. It permitted the extraction of the full-text content from the record’s metadata and also the INFTY results for each of those contents.

Together, all these components create a unified harvest workflow that allows for future harvesting of new data to be done continuously and with little effort.
7. References

