

DELIVERABLE

Project Acronym: CARARE
Grant Agreement number: 250445
Project Title: *Connecting ARchaeology and ARchitecture in Europeana*

D3.3.4 Briefing paper on metadata mapping and the use of mapping tools

Revision: [1]

Authors:

Vassilis Tzouvaras, Kostas Pardalis, Arne Stabenau, Fotis Xenikoudakis and Nasos Drosopoulos (NTUA)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	√
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organisation	Description
0.1	20/02/11	Vassilis Tzouvaras, Nasos Drosopoulos	NTUA	First Complete Draft
0.2	04/03/11	Vassilis Tzouvaras, Nasos Drosopoulos	NTUA	Final Draft taking into account the reviews from Kate Fernie, Rimvydas Laužikas, and Christian Ertmann-Christiansen

Table of Contents

EXECUTIVE SUMMARY	4
INTRODUCTION	5
1. CARARE AGGREGATION	7
Ingestion workflow	7
Content upload	8
User and organisation management	8
Functionality and user interfaces	9
2. THE MAPPING MODULE	18
Mappings	19
3. STATISTICS	25
4. TRANSFORMATION SERVICES	27
Transform	27
Review Transformed Dataset	28
5. IMPLEMENTATION DETAILS	29
Functional analysis	30
6. RELATED WORK	32
The HP-MIT DSpace Repository project	32
The Fedora Digital Object Repository Management System	34
The EPrints repository platform	35
The CERN Document Server Software (CDSware)	36
DRIVER: Building a sustainable infrastructure of (European) Scientific Repositories	36
REPOX – A Metadata Space Manager	38
7. CONCLUSIONS	38
8. REFERENCES	39

Executive Summary

This deliverable reports on the design, development and deployment of the software services for the ingestion of cultural content metadata originated from providers that participate in the CARARE project. The discussed workflow aims to guarantee semantic interoperability across the different repositories of varied technical features, capabilities and scope, allowing seamless ingestion of diverse content and knowledge. The core modules that currently serve the ingestion process are described, together with examples and screens that highlight important functionalities of the implemented web services. This document can be used as a reference for the prototype design and implementation as well as an instruction manual to familiarize with the services' features and usage.

The rest of this document is structured as follows:

Chapter 1 describes the CARARE aggregation service that is responsible for setting the environment, user roles and access rights to define the tasks that users can perform on specific organisations' repositories. It is the basis of the whole service and gives access to the functionality of the system. Chapter 2 outlines the Mapping module as it is set up to support CARARE Metadata Schema, together with relevant functionalities that enable semantic interoperability for the ingested content. Chapter 3 gives an overview of the Statistics module and relevant information and functionality that are presented to the user in order to guide the mapping process. Chapter 4 describes the transformation services. Chapter 5 provides the implementation details. Chapter 6 contains an overview of relevant platforms and tools that deal with ingesting, mapping and transforming metadata records as well as with enabling permanent access to digital works. Finally, Chapter 7 summarizes the conclusions of the report. Chapter 8 presents the references.

Introduction

The CARARE ingestion workflow is established to support the consortium's needs through close cooperation with all relevant work packages. It has been implemented on the metadata interoperability platform of NTUA which has been customised and extended to support the project's requirements. The primary focus is to support the aggregation of the various provider organisation data models, using the CARARE schema as the reference metadata model, and establishing semantic, machine understandable crosswalks for providers' datasets. The basic steps in the process that lead to semantic interoperability and allow for the ingestion and aggregation of all cultural heritage content within CARARE, and the subsequent publishing of the semantically interoperable metadata, especially for harvesting by the Europeana portal, are as follows:

- registration and access rights for users and their respective organisations, also available for supporting regional aggregators;
- import and parsing of organisations' metadata records, supporting any proprietary or standardised schema;
- analysis (statistical, structural and semantic) of user input in order to provide a detailed overview and to assist the user in the subsequent steps with previewing and guiding capabilities;
- cross-walk editing and transformation of user metadata records to a reference, well defined schema that will allow for bidirectional interoperability with all standardised outside sources.

The creation of the CARARE Metadata Schema, coupled with the loosely defined providers' input schemas, lead to an aggregation and mapping workflow that allows for an elaborate, visually guided ingestion of metadata in the repository. The fundamental principles include the disassociation of input metadata from existing metadata standards in order to avoid ambiguity over interpretation and the ability to create and manage transformations that apply to the actual metadata records, which subsequently (re-)define the input schema in a semantic, machine understandable way based on its mapping to the CARARE Metadata Schema.

The CARARE ingestion tool provides a user friendly environment that allows for the extraction and presentation of all relevant and statistical information concerning input metadata together with an



intuitive mapping service for the CARARE schema, and provides all the functionality and documentation required for the providers to define their crosswalks. Transformations are editable and reusable and can be applied incrementally to user input while providing, throughout all steps, best practice examples, previews and visual indications to illustrate and guide user actions. One of the key capabilities lies in the ability to semantically enhance user metadata through conditional mapping of input elements and use of value transformation functions (e.g. string manipulation) that will allow for the addition and enrichment of records even with metadata that are not present in the input.

1. CARARE Aggregation

CARARE has set up a platform to offer services for content providers in order to perform ingestion and aggregation of resources, leveraging the expertise and available resources regarding metadata crosswalks. Due to the nature of a European thematic aggregation there is an expected diversity among participating providers and content. The software tools need to accommodate inexperienced users and legacy data, while taking advantage of the domain experts' knowledge and the project's working groups' results. A tool for visually mapping local metadata schemas to the CARARE schema helps ensure the success of a large scale aggregation by providing an intuitive, user-friendly approach that reduces the effort to create translation logic for mappings and the turnaround time between human-readable crosswalks and executable code.

Ingestion workflow

CARARE project's ingestion procedure illustrated in Fig. 1, consists of four phases, each responsible for specific services required in order to ensure the effectiveness and quality of the aggregation.

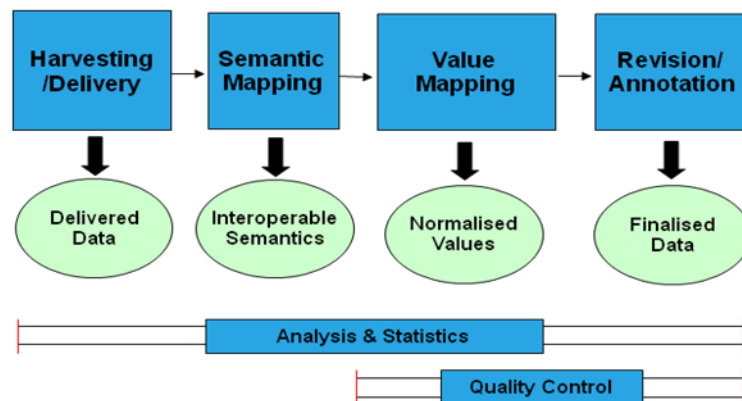


Figure 1. Ingestion workflow

Harvesting/delivery is the process responsible for collecting the metadata from distributed repositories. It will be an interface for different methods of data delivery such as HTTP or FTP upload and OAI-PMH.

Semantic Mapping will provide the service for assigning semantics to the harvested metadata. A mapping tool will assist providers to manually align their local schemas to the reference data model.

Providers that have metadata in supported known formats might be able to omit this step by using stored transformations from selected schemas to the reference schema based on existing crosswalks.

Value Mapping ensures the correct formulation of values for controlled metadata fields. It will enable providers to resolve data issues, e.g. map own terminology list to selected terminology lists and to automate data normalization according to selected vocabularies and best practices for values such as dates, geographical locations, nationality/language, etc.

Revision/Annotation will enable the revision of the transformation results as well as the editing or addition of data that is not in the original metadata (e.g. empty fields or, ones that take values from controlled vocabularies).

Across the four phases, a set of tools for **Analysis & Statistics** provides detailed information on the metadata contributed by a provider (i.e. number of items imported, total values per field etc), while **Quality Control** procedures will automatically check and report on ingested metadata (i.e. missing values, malformed data).

Content upload

Currently the data formats allowed for uploads are:

- XML in any schema.
- zip archives of the above

The following methods are supported for uploading content:

- HTTP upload; suggested only for relatively small amounts of data (<2MB)
- Upload to a dedicated FTP server.
- Remote HTTP or FTP browsing.
- OAI-PMH repository harvesting.
- SuperUser uploading from local file system (restricted).

User and organisation management

Users belong exclusively to one organization and cannot access data not related to that organisation. Users can be assigned with different levels of access, that grant roles ranging from data browsing,

over editing and annotating, to being allowed to edit other users' details (administrators). We have extended user roles to allow parent users, for organizations that might not have expertise or manpower to use the system and thus, delegate the job to an organization which is then their designated "parent" organization. Parent users extend their rights to child organizations and provide the functionality to build the access hierarchy for any given country/thematic category. The current role set can be easily adjusted to allow more freedom in rights management.

The following rights are currently implemented:

- change/add/delete user
- change/add/delete organization
- edit/upload/delete data
- publish / declare finished datasets
- read-only browsing rights

These rights have been grouped to the following roles:

- Administrators (all user, data and organization rights for the organizations they manage),
- Annotators (data management rights),
- Publishers (publishing data rights),
- Data Viewers (simple viewing rights).
- The system also contains some hardcoded super-users that have full rights to all organizations and their data in the system.

Functionality and user interfaces

Users can join the CARARE service using the registration page (fig. 1.1). During registration they are prompted to select the organization they belong to.

LOGIN REGISTER

Register

Username*:

Password*: No password

Password Confirmation*:

First Name*:

Last Name*:

Email*:

Contact phone num:

Job role:

Organization:

powered by [mint](#) - ©National Technical University of Athens

Figure 1.1 Registration Screen

The administrator of that organization is notified by email for the pending user registration and is authorized to grant the appropriate rights and finalize the procedure.

In case a user's organization is not present in the list of registered organizations, the user can register in the system without providing one. In this scenario he is given the opportunity to create a new organization and automatically become the administrator for it.

HOME
MY PROFILE
OUTPUT XSD
ADMINISTRATION
IMPORT
OVERVIEW
LOGOUT

CARARE Ingestion Server

You are currently logged in as user **admin** (role: **superuser**)

READ latest

Server statistics

55 registered users / 29 organizations from 18 countries.
 35862 imported items
 3858 transformed items

ver. 2co90

User roles:

- Administrator: This user can create/update/delete users and children organizations for the organization he is administering. He/she can also perform uploads and all available data handling functions provided by the system.
- Annotator: This user can upload data for his/her organization (and any children organizations) and perform all available data handling functions (view items, delete items, mappings etc) provided by the system, apart from final publishing of data.
- Annotator & Publisher: This user has all the rights of an annotator as well as rights to perform final publishing of data.
- Data Viewer: This user only has viewing rights for his organization (and any of its children organizations).
- No role: A user that has registered for an organization but has not yet been assigned any rights.

Registered organizations:

- 1Spatial-ADS (United Kingdom)
- AVINET (International)
- Cultural and Educational Technology Institute (Greece)
- Cultural Heritage Agency (Netherlands)
- Cyprus Research and Education Foundation - The Cyprus Institute (Cyprus)
- DANS-KNAW (Netherlands)
- Digital Curation Unit-IMIS, Athena Research Centre (Greece)
- Directorate of the National Archive of Monuments (Germany)
- Europeana (Europe)
- European Center of Byzantine and

Figure 1.2 Home screen

Organizations within the system can have parental organizations. Users of parental organizations extend their rights to the children of the organization (and in turn to grandchildren that may exist, and so on). Every organization can have at most one parent organization. The parent organization has to agree on publishing the data of the child organizations (among other things). This way an aggregator for example can define and manage all the organizations (and their respective data) he is supervising within CARARE.

Administration

Select a user login to view all the user details:

		Login: ekt_nh	Name: Houssos, Nikos
		Login: VTM	Name: Autere, Riitta
		Login: AviR	Name: Rosenberg, Avi
		Login: mmb_ma	Name: Anastasiou, Marianna
		Login: npapad	Name: Papadimitriou, Nikolas
		Login: lido_su	Name: su, lido
		Login: SEmuzej	Name: Sosič, Barbara
		Login: nicola.s	Name: Stoyanov, Nicola
		Login: ntng_dv	Name: Valeonti, Dimitra

Select an organization to view all its details:

		Name: Test_organization2
		Name: Test_organization3
		Name: Test_organization14
		Name: Test_organization15
		Name: Test_organization6
		Name: Test_organization5
		Name: Test_organization7
		Name: Test_organization8
		Name: Test_organization9

powered by mint - ©National Technical University of Athens

Figure 1.3 User's Administration Screen

Every user of the system has the right to see all the other users and data within the same organization (fig. 1.3). S/he can change her own details by using the Profile page (fig. 1.4).

My profile

Your registered details follow. Click on "edit details" to update them:


Registered info	
Username:	admin
First Name:	carare
Last Name:	Admin
Email:	stabenau@image.ntua.gr
Contact phone num:	
Organization:	NTUA
Job role:	
System role:	superuser
Account created:	1/1/09 12:00:00 AM.000
 Edit details	

Figure 1.4 User Profile Screen

Administrators and Annotators of an organization can upload data using the import page (fig. 1.5).


Import

Select your import method:


Http Upload Only zip and xml files allowed


NTUA FTP Upload NTUA FTP:

Remote FTP/HTTP Upload Give URL to remote ftp/http server

OAI URL Give link to OAI repository  check oai url

From Date (YYYY-MM-DD): To Date (YYYY-MM-DD):

OAI SET:  fetch OAI sets

Namespace Prefix:  fetch OAI namespaces

Server filename Server file path for upload

Upload for Organization*: Parent organization upload support

Figure 1.5 Import Interface

A history of all uploads for an organization can be browsed using the Overview interface (fig. 1.6). Different icons are used to show the current status of an import (hourglass for processing, green arrow for completed, red 'x' for failed).

An import can be deleted, thus deleting all items it contains. After an import process has been completed, mappings have to be defined for the data set in order to see all the items it contains. Every mapping defined for an organization can be saved, edited and reused at a later stage.

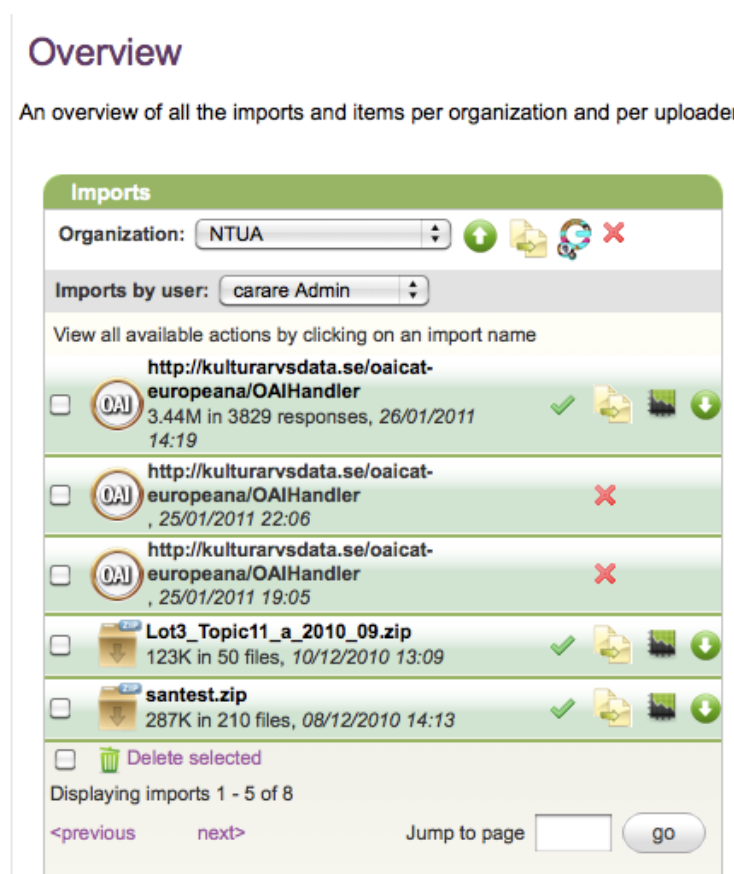


Figure 1.6 Overview Interface

In the overview screen the user can browse through items that have been uploaded. Metadata can be uploaded in a single XML containing multiple items or in multiple XML files, each one containing one item. In order to preview the items that belong to each upload, the user has to firstly define the item's root element in the XML structure (fig. 1.8) and additionally an element that will serve to label the separated items. The actions that are available to the user as part of the Overview Interface are the following and presented in (fig. 1.7):







- When a “Green” tick appears, near the import name, it indicates that the importing process was successful. If a “Red” cross appears it means that the importing process has failed. In any case if the user hovers the mouse over the icon; various information regarding the process is displayed.
- . This is the Show Items icon. When the user presses this button a new modal window is entered where he/she is able to review the dataset of the import. More details about this functionality in the “Review Original Dataset” chapter.
- . This is the statistics button. When the user presses it a new modal window is rendered where various information regarding the imported dataset is presented. All the different XPathS that were extracted are presented in a tabular form together with statistical information regarding the distribution of the various values of each XPath.
- . The download button. When the user presses it he/she is able to download an archive with all the XML files that are part of the ingested dataset.



Figure 1.7 How the imports are presented to the user in the “Overview” Tab.

When the user clicks on the name of the Import an extended view for the current Import is presented as depicted in (fig 1.9). The rest of the mandatory steps that are part of the ingestion tool core workflow are executed from this extended view. More specifically:

- . This button invokes the process for defining the root and label element from the extracted XPathS from the ingested data set.
- . This button executes the transformation of the items.
- . This button invokes the mapping tool in order to perform the semantic mappings between the source and target Schemas and produce an XSLT.

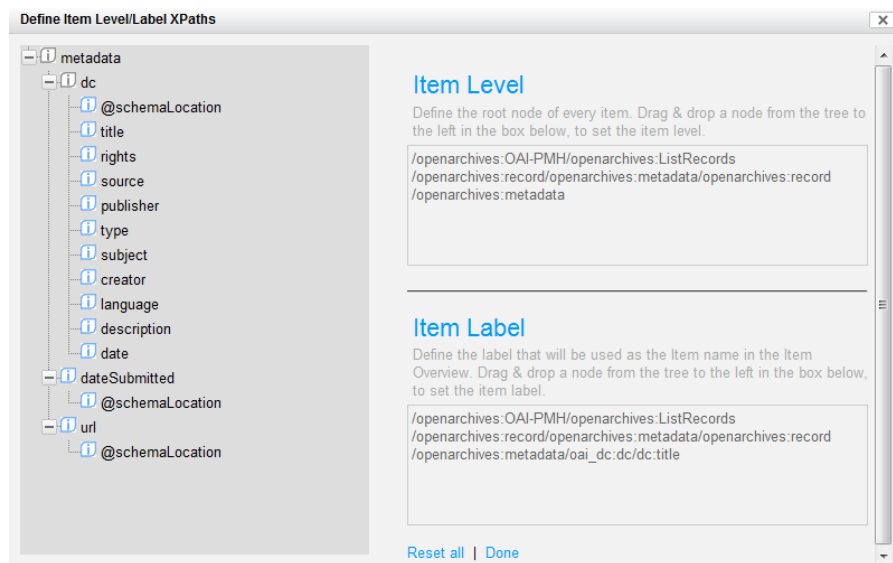


Figure 1.8 Definition of item root element

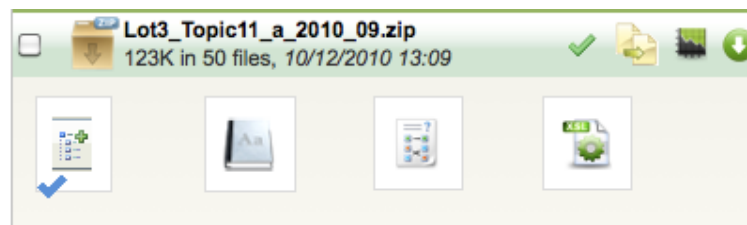


Figure 1.9 The extended view of an Import in the "Overview" Tab.

Having defined the item root element, the items that belong to the specific upload can now be previewed (fig. 1.10). Finally, through an icon next to each item, the user views the original XML (fig 1.11)

Items	
(carare Admin, NTUA)	
Filter by: Lot3_Topic11_a_2010_09.zip	
Name	Created
Barbara	.._2010_09.zip 10/12/2010 13:09
Carte de Séjour, "Douce France"	.._2010_09.zip 10/12/2010 13:09
Interview with Louis-Ferdinand Céline	.._2010_09.zip 10/12/2010 13:09
Picasso at Vallauris	.._2010_09.zip 10/12/2010 13:09
Albert Camus	.._2010_09.zip 10/12/2010 13:09
The cinema fortnight	.._2010_09.zip 10/12/2010 13:09
NTM, "Qui paiera les dégats ?" and "Police"	.._2010_09.zip 10/12/2010 13:09
Marguerite Duras talks about her literary style	.._2010_09.zip 10/12/2010 13:09
Brigitte Bardot on filming "La Vérité"	.._2010_09.zip 10/12/2010 13:09
Léo Ferré	.._2010_09.zip 10/12/2010 13:09

Displaying items 1 - 10 of 50

<previous next> Jump to page go

Figure 1.10 Items Screen

```

XML Preview - Input
Input XML
view plain print ?
01. <?xml version="1.0" encoding="UTF-8"?>
02. <euscreen:EUScreen xmlns:euscreen="http://www.euscreen.eu/schemas/euscreen/">
03.   <euscreen:metadata>
04.     <euscreen:AdministrativeMetadata>
05.       <euscreen:provider providerCode="12">INA</euscreen:provider>
06.       <euscreen:publisherbroadcaster>ORTF</euscreen:publisherbroadcaster>
07.       <euscreen:iprRestrictions>true</euscreen:iprRestrictions>
08.       <euscreen:rightsTermsAndConditions>Copyright : Institut national de l'audiovisuel (www.ina.fr). All
09.     </euscreen:rightsTermsAndConditions>
10.     <euscreen:firstBroadcastChannel/>
11.     <euscreen:identifier>EUS_00A95427D0ED46869ED9D583736D4674</euscreen:identifier>
12.     <euscreen:uri>Ina_I06013650_J-04733.mp4</euscreen:uri>
13.     <euscreen:originalIdentifier>Ina_I06013650_J-04733</euscreen:originalIdentifier>
14.     <euscreen:filename>Ina_I06013650_J-04733.mp4</euscreen:filename>
15.   </euscreen:AdministrativeMetadata>
16.   <euscreen:ContentDescriptiveMetadata>
17.     <euscreen:TitleSet>
18.       <euscreen:TitleSetInOriginalLanguage>
19.         <euscreen:title>Barbara</euscreen:title>
20.         <euscreen:seriesTitle>CHANSONS POUR UNE CAMERA</euscreen:seriesTitle>
21.       </euscreen:TitleSetInOriginalLanguage>
22.       <euscreen:TitleSetInEnglish>
23.         <euscreen:title>Barbara</euscreen:title>
24.         <euscreen:seriesTitle>CHANSONS POUR UNE CAMERA</euscreen:seriesTitle>
25.       </euscreen:TitleSetInEnglish>
26.     </euscreen:ContentDescriptiveMetadata>
27. </euscreen:EUScreen>
  
```

Figure 1.11 Input XML

2. The Mapping Module

The core module of the CARARE ingestion tool is the mapping tool. Although the service shares functionality with many existing metadata repositories, e.g. DSpace and Fedora, one of its main goals is to provide support for a great diversity of metadata schemas or simple data structures, thus widening metadata interoperability. The CARARE Ingestion platform aims to be able to store and manipulate metadata that are described using different conceptual models for encoding and decoding information. For this reason both a syntax and semantics have to be defined in order to obtain a complete and expressive model. XML is used as the machine understandable syntax and can be interpreted using different parsers depending on the specified needs. The mapping tool provides the interfaces and mechanisms for identifying and registering through a reference model the semantics of the models used.

Data integration processes comprise of various tasks including data matching, data transformation, and schema/semantic matching. Many solutions have been proposed by the community for each one of those tasks, ranging from applications that rely heavily to the user, to applications that are semi-automatic and in some cases completely automatic depending on the task, thematic category and schema complexity. For the case of schema/semantic matching many techniques and platforms have been developed, enabling the user to complete successfully the task. Notable cases are the schema mapping tool provided by Altova that offers a rich editing environment where the user is able to map any number of arbitrary schemas, but the whole process is totally manual, and the COMA++ platform that offers the user an environment for semi-automatic schema mapping, using state of the art algorithms. Although these approaches attempt to solve the general problem, the case of the CARARE project has specific characteristics that lead to a more specialized solution to efficiently handle large amounts of diverse data and metadata.

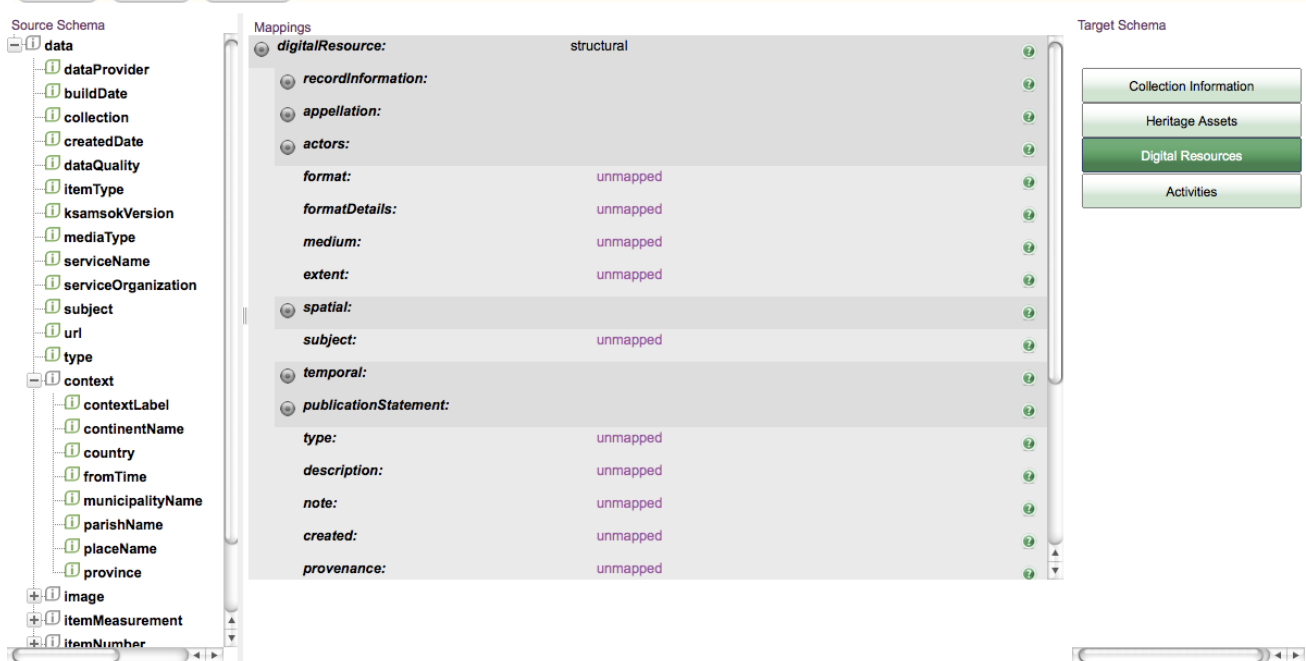
By choosing a semantically rich and well-defined reference schema as the target of the mapping process the user has the opportunity to semantically enrich his data and metadata while quality of the aggregated content is ensured. The mapping process is manual and the tool offers previewing, assisting and validation capabilities in order to ensure the quality of the result. The design principles of the mapping tool ensure the extensibility of the tool itself on a software level and of the system on a data level.

Mappings

The mapping tool allows the user to define semantic mappings between the source and target schemas. An XSLT is then generated based on these mappings that can automatically convert all imported items. An example of the mapping tool is shown in the following is depicted in (fig 2.2).

Mappings: deli

Define your mappings and when you are done click the 'Finished' button below to make them available to the rest of the users in your organization.
 *Mapping relations are automatically saved every time you edit, delete or create a new one.



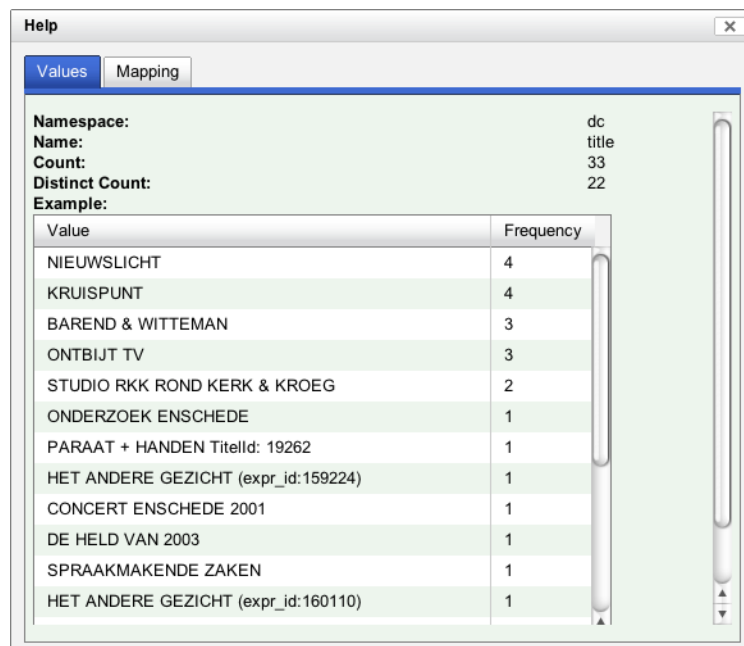
The screenshot shows the Mapping tool interface. On the left is the 'Source Schema' tree with nodes like 'data', 'context', 'image', etc. In the center is the 'Mappings' table for the 'digitalResource' structural element, listing various properties like 'recordInformation', 'actors', 'spatial', etc., all marked as 'unmapped'. On the right is the 'Target Schema' with buttons for 'Collection Information', 'Heritage Assets', 'Digital Resources', and 'Activities'.

Property	Value
format:	unmapped
formatDetails:	unmapped
medium:	unmapped
extent:	unmapped
subject:	unmapped
type:	unmapped
description:	unmapped
note:	unmapped
created:	unmapped
provenance:	unmapped

Figure 2.2 Mapping tool

Source schema

On the left of the mapping tool is a tree like structure of the source schema similar to the one presented during dataset item level selection. The user can navigate in the schema by clicking the nodes on the left of the tree elements. Elements with a grey information icon (i) are structural elements and do not contain data values. The rest of the elements are leaf elements with data while elements starting with '@' are attributes of the corresponding father. Elements in blue are elements that have been already used in this mapping. More information about the schema elements is provided by clicking the i icon. This action invokes a panel as show in the following Figure:



Value	Frequency
NIEUWSLICHT	4
KRUISPUNT	4
BAREND & WITTEMAN	3
ONTBIJT TV	3
STUDIO RKK ROND KERK & KROEG	2
ONDERZOEK ENSCHEDE	1
PARAAT + HANDEN Titellid: 19262	1
HET ANDERE GEZICHT (expr_id:159224)	1
CONCERT ENSCHEDE 2001	1
DE HELD VAN 2003	1
SPRAAKMAKENDE ZAKEN	1
HET ANDERE GEZICHT (expr_id:160110)	1

Figure 2.3 Source schema element information panel

This panel contains two tabs: Values and Mapping. The Values tab shows the following information about the specific element:

- **Namespace:** The XML namespace to which this element belongs.
- **Name:** The element name.
- **Count:** The number of times the XPath of this element exists in the imported dataset.
- **Distinct Count:** The number of unique values associated with this XPath in the imported dataset.
- **Example:** A sample of these values sorted by their frequency of appearance in the imported dataset.





The Mapping tab shows where and how the specified element is being used in the mapping that is being edited.

Target Schema & Mapping Area



The target XML schema is split into sections that appear as buttons on the right side of the mapping tool. These buttons are used to navigate to specific parts of the target XML schema. By clicking these buttons, the corresponding part is loaded in the middle of the mapping tool along with its specified mappings.



Figure 2.4 Example of the mapping definitions

Each row in the mapping area corresponds to a mapping element of the target xml schema. Rows with grey background are structural elements and contain other elements. They can be expanded and their children can be seen by clicking the  button to left of their name. The rest of the elements can contain data and have an 'unmapped' area which can be used to define mappings, as explained later. Elements can also have attributes which can be seen by clicking the  icon when available. If an element can exist more than one time in the target XML file then the  icon is available on the right of the element row. By clicking this icon an additional row will appear on the mapping area. More information about each element can be provided by clicking the corresponding  icon on the right of the element row. Elements with green names are elements with mappings or have children with mappings. Elements with red names are mandatory elements that have no defined mappings or have mandatory children with no mappings defined.

Mapping types

Various mapping types are available by using the mapping tool. The most common type of mapping is **XPath mapping**. This type of mapping will copy data from a source element to a target element. To define this kind of mapping, drag n' drop a source element to the 'unmapped' area of a target element. The source element will then appear in the unmapped area of the target element. By clicking the  icon on the left of the source element name, an additional unmapped area will appear for the same target element. More mappings can be performed on this unmapped area and the XML result will contain a concatenation of the provided values. Clicking the corresponding  icon will remove a defined mapping.

Double clicking on the unmapped area will define a **constant value mapping**. The following panel is invoked:

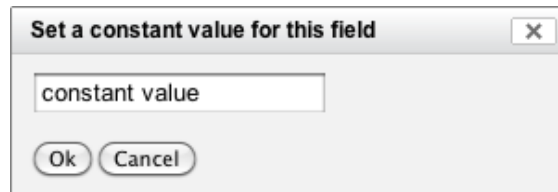


Figure 2.5 Constant Value Panel

The user can type a constant value in the provided text field. The value will then appear in the mapping area and in the result XML files. This type of mapping is useful for text that is intended to appear in all transformed items. Constant value mappings can be combined with XPath mappings to construct specific values such as URLs.

Conditions

Mappings can be restricted so that they will apply only under certain conditions. To define these conditions the ★ button is used. This will allow the input of condition as shown in the following figure:

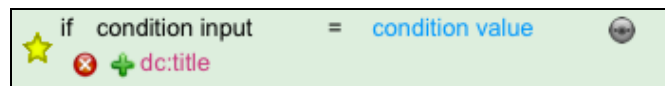


Figure 2.6 Mapping Condition

The conditions supported are in the form of <Source XPath> = <Constant Value>. The corresponding mapping will apply only if the source XPath data for a specific item equals to the constant value provided. The condition source XPath can be set by dragging n' dropping a source element to the condition input area as shown in the previews Figure. The constant value can be set by double clicking on the constant value area.

If a more complex condition is required then the provided condition editor must be used by clicking on the ⚙ icon next to the condition. An example of the condition editor is shown in the following figure:

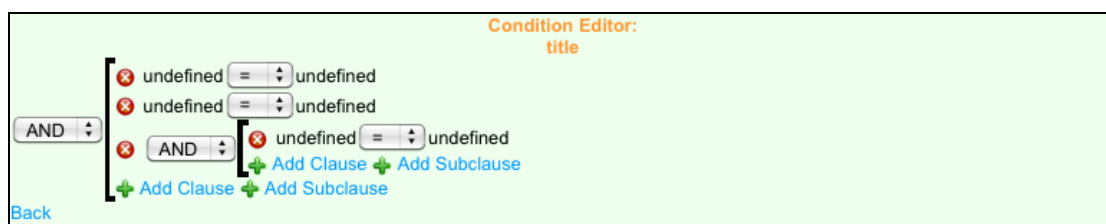
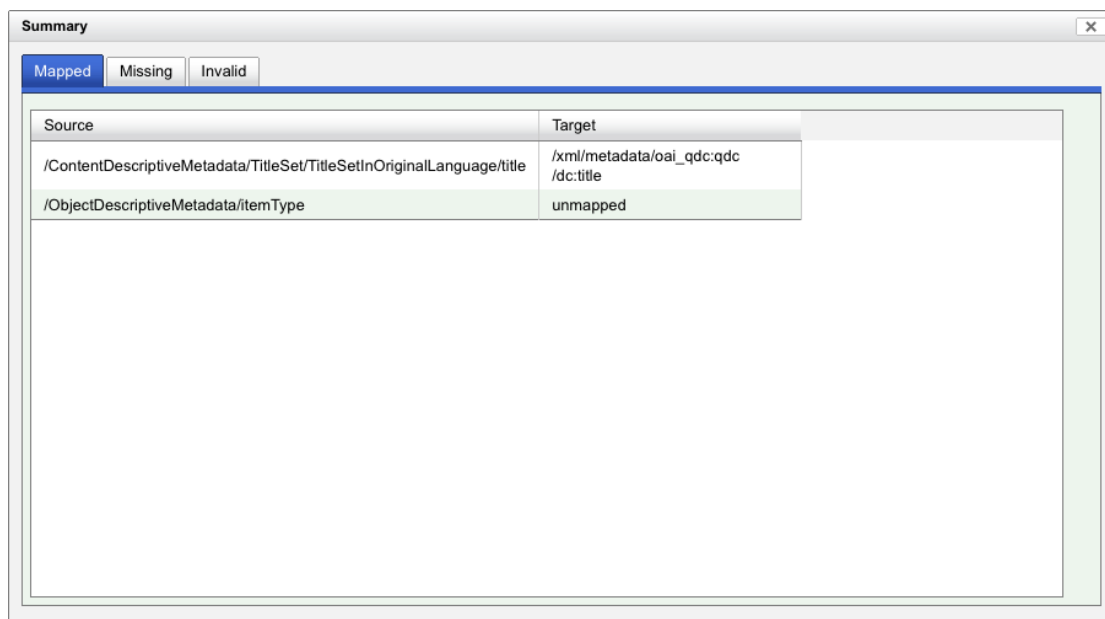


Figure 2.7 Condition Editor

The condition editor provides the means to define more complex conditions. In addition to each clause's source element and constant value, the relational operator can be set. The logical operator that combines the clauses can also be defined. Additional clauses or subclauses can also be created as needed, by clicking the corresponding **+** icon.

Preview & Mapping Summary

A summary of the defined mappings can be seen by clicking the 'Summary' button on the top of the mapping tool. This will invoke the following panel:



The screenshot shows a window titled 'Summary' with three tabs: 'Mapped', 'Missing', and 'Invalid'. The 'Mapped' tab is selected and displays a table with two columns: 'Source' and 'Target'.

Source	Target
/ContentDescriptiveMetadata/TitleSet/TitleSetInOriginalLanguage/title	/xml/metadata/oai_qdc:qdc /dc:title
/ObjectDescriptiveMetadata/itemType	unmapped

Figure 2.8 Mapping Summary Panel

This panel contains the following tabs:

- **Mapped:** All mapped source elements and the corresponding target elements.
- **Missing:** Mandatory target elements that have no mappings.
- **Invalid:** If a mapping definition was loaded based on another dataset, all XPaths from this dataset that do not exist in the current dataset appear in this tab.

The generated XSL can be previewed at any time by clicking the 'Preview' button on the top of the mapping tool. This will invoke the following panel:

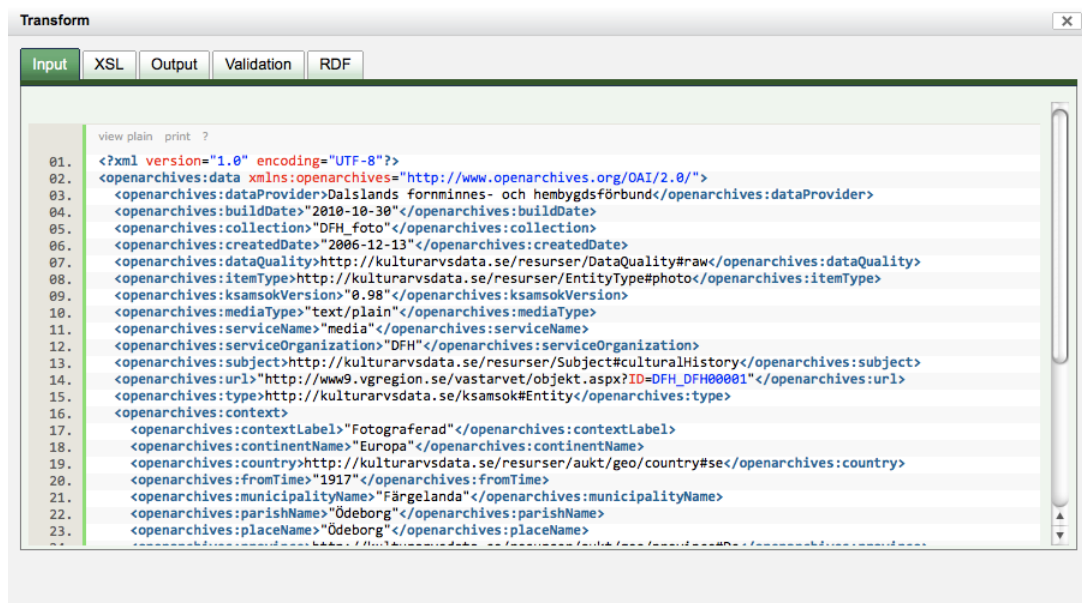


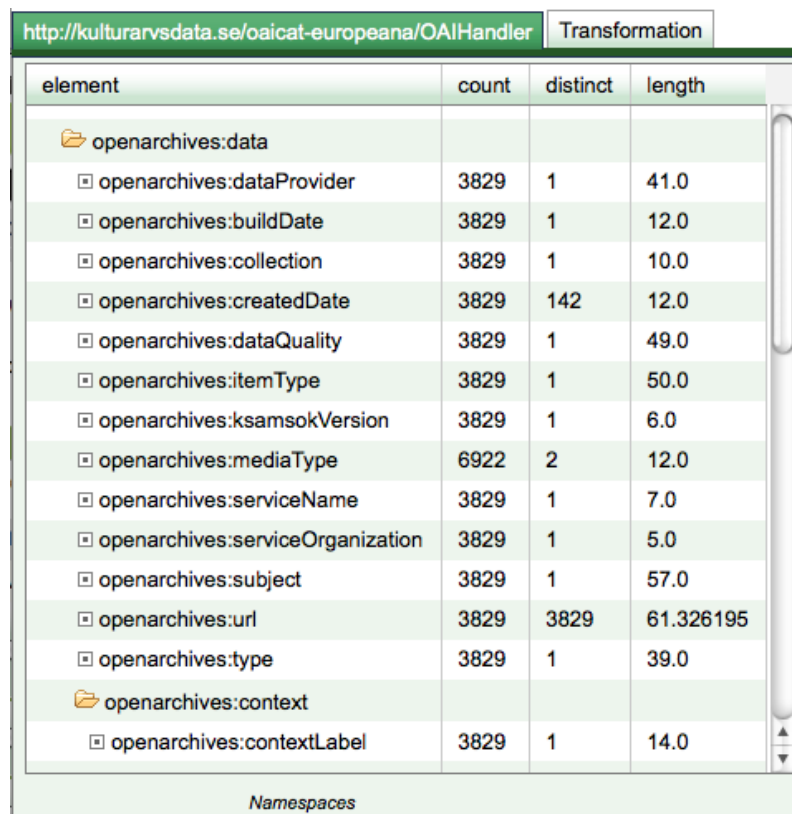
Figure 2.9 Preview Transform Panel

This panel contains the following tabs:

- **Input:** The XML that corresponds to the first item from the imported dataset.
- **XSL:** The generated XSL.
- **Output:** The item transformed to the CARARE schema based on the previous XSL
- **Validation:** Validation information for the transformed item. Any errors from wrong or incomplete mappings will be reported there.
- **RDF:** The output in EDM RDF.

3. Statistics

The statistics module facilitates the mapping and monitoring procedures. The following figure illustrates the statistics interface.



element	count	distinct	length
openarchives:data			
openarchives:dataProvider	3829	1	41.0
openarchives:buildDate	3829	1	12.0
openarchives:collection	3829	1	10.0
openarchives:createdDate	3829	142	12.0
openarchives:dataQuality	3829	1	49.0
openarchives:itemType	3829	1	50.0
openarchives:ksamsokVersion	3829	1	6.0
openarchives:mediaType	6922	2	12.0
openarchives:serviceName	3829	1	7.0
openarchives:serviceOrganization	3829	1	5.0
openarchives:subject	3829	1	57.0
openarchives:url	3829	3829	61.326195
openarchives:type	3829	1	39.0
openarchives:context			
openarchives:contextLabel	3829	1	14.0

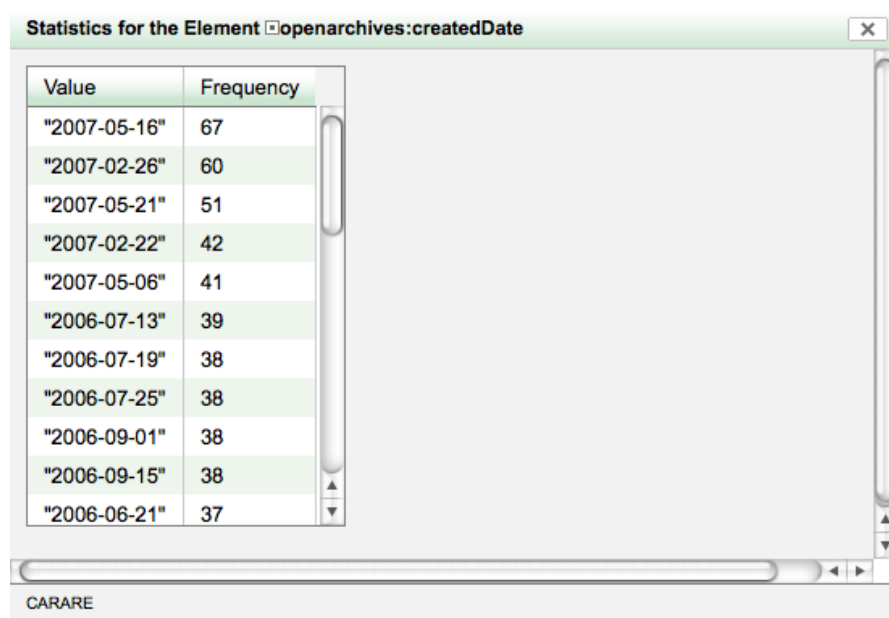
Namespaces

Figure 3.1 Input schema statistics.

- **Element.** This is the name of the Element or the attribute of an element found in the import and that belongs to a specific XML Schema.
- **Count.** The number of times this element appear in the upload
- **Distinct.** The numbers of distinct/unique values the element or attribute holds.
- **Length.** The average length of the values the element or attribute holds.

Figure 3.2 present the interface of the values of an element.

- **Value.** The rows under this column contain a distinct value found for a specific element or attribute.
- **Frequency.** The rows under this column contain the frequency of that specific value that appears in the preceding row.



Statistics for the Element openarchives:createdDate


Value	Frequency
"2007-05-16"	67
"2007-02-26"	60
"2007-05-21"	51
"2007-02-22"	42
"2007-05-06"	41
"2006-07-13"	39
"2006-07-19"	38
"2006-07-25"	38
"2006-09-01"	38
"2006-09-15"	38
"2006-06-21"	37

CARARE

Figure 3.2 Value distribution statistics for a specific element or attribute.

4. Transformation Services

Transform

When the user has successfully defined the root and label elements for a specific “Import” and the mappings between the extracted source XML Schema and the target Schema of the system, he/she is able to perform the transformation of the data. For this to happen the user has to click the  button which is visible under the extended view of a specific “Import” in the “Overview” tab. When this event is triggered by the user, he/she is presented with the modal window depicted in (fig. 4.1). The user is presented with information regarding the different states a mapping might be based on the appropriate Icons. In order for the user to continue the transformation process he/she has to select a mapping from the drop down list and click on the “Submit” button.

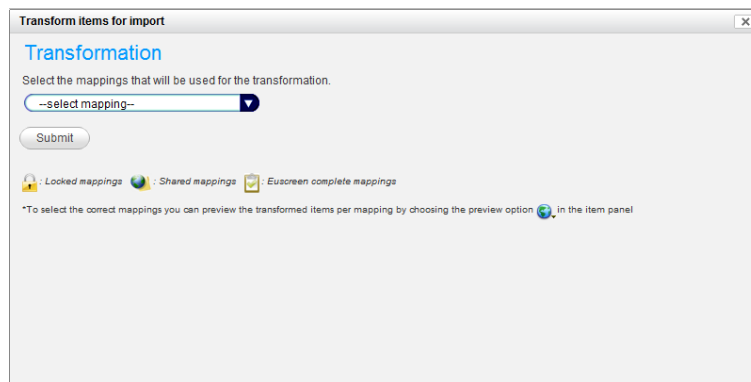


Figure 4.1 The transformation modal window.

In the case where the mapping is not correct, for example mandatory mappings are missing, the user is presented with a modal window explaining what the problems are, like the one depicted in (fig. 4.2). This modal window has two distinct tabs presenting different kind of information to the user. The first tab presents any missing mappings to mandatory XPathS of the target Schema while the second one presents XPathS with erroneous mappings. The user is able to review the errors and then he/she has to go back and either select a different mapping or complete/correct the current select one.

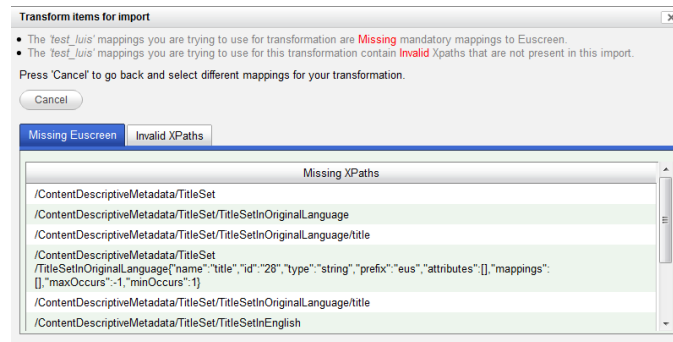





Figure 4.2 the modal window presenting the errors found in a selected for transformation mapping.

If everything goes well and the defined mappings do not contain any invalid XPathS or and there are no Missing mandatory mappings the user is redirected back to the “Overview” tab and an animated Icon takes the position of the  button. When the user positions the mouse pointer over the animated icon, information regarding the progress of the transformation process is presented to him on a tooltip. Actually, while in the process of transformation, the system extracts each item XML instance based on the root element the user has defined and applies the XSLT transformation that is generated through the process of defining the mappings between the two Schemata, every generated item is then stored to the ingestion tool persistent data layer and is associated with the current Import. In the case where an error occurs in the process of transformation a “Red X” icon appears on top of the  icon. When the user positions the mouse pointer on top of the icon he/she is able to review the errors that caused the transformation process to abort. In the case where the transformation ends without errors a “Blue” tick sign appears on top of the “transformation” icon and the user is able if he/she wishes to download the transformed items.

Review Transformed Dataset

After the completion of the transformation step in the ingestion tool core workflow, the user is able to review the original data together with the resulted transformed items and the generated XSLT on a per item level through the item browser in the “Review” tab. In order to do that the user has to press the  for an individual item in the item browser. When this event occurs the user is prompted with a modal window where he/she is able to review the results of the whole process as depicted in (fig. 4.3). The user is able to view the Input XML, the Invalid XPathS if any exist, the XSL generated in the mapping process, the output XML in the target Schema of the system and the output XML of the Publishing Schema.

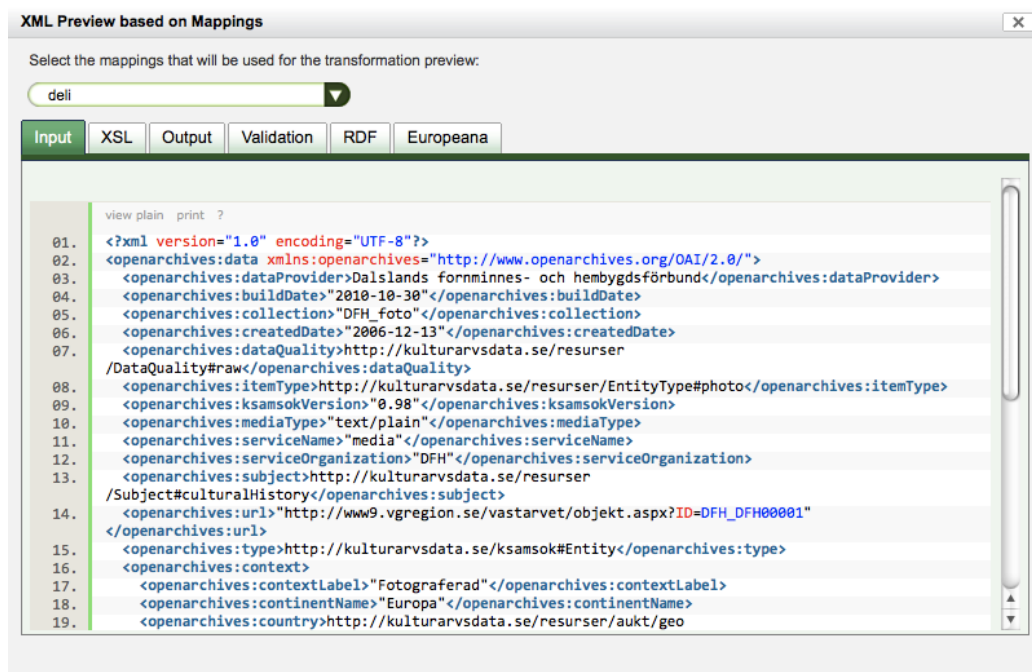


Figure 4.3 The modal window where the user is able to review the results of a transformation on an individual item.

5. Implementation details

The system is implemented as a web service, where authentication is required to perform a series of tasks that correspond to work flow steps. The service is an application written in the Java programming language and hosted on a web server by the Tomcat servlet engine. Data is imported into a PostgreSQL database in xml format (as BLOB).

Once uploaded, the xml structure is parsed and represented in a relational database table. As this table can grow quite large it is partitioned into one partition per data upload. All data within one upload is treated as having the same structure, so it is not possible to upload different schemas (or more likely updated schemas) in one upload.

Most of the communication between the application and the database is implemented on the Hibernate framework, a high performance object/relational persistence and query service. This allows for powerful, yet simplified, management of housekeeping objects like Users, Organizations and Data Uploads while also providing additional functionalities such as integration with Lucene for indexing and querying data.

Once data is parsed into the relational table, indexes are built to allow quick access to any part or sub-tree of the xml-tree like data. These are currently constructed as PostgreSQL BTREE indexes; when content full text indexing is implemented, it will be based on Hibernate's search architecture. All further data manipulation such as mapping and transformation, normalization, enrichment, etc. is structured through the addition of extra tables annotating but not altering the original data. This allows easier comparison between uploads and facilitates the versioning strategy.

Functional analysis

The mapping tool is designed using a client – server approach. A subset of the functionalities is implemented as server side services, while the user interface is rendered on the client inside a web browser. The communication between the client and the server is achieved using AJAX calls. One of the core design concepts of the mapping tool is that the user should be able to use all the functionality he might need in order to achieve the best possible result with minimal effort. In order to achieve that, the tool must be intuitive and visual aiding and appealing. Another important design concept is that performance must be ensured because the Ingestion platform must be able to perform computational intense tasks, e.g. metadata transformation and data parsing, without affecting the interaction between the user and the web service. This is achieved in a great degree by separating the interface rendering and the interaction with the user from intense tasks that are executed on the server side. At the same time the communication overhead between the client and the server was minimized as much as possible.

The XML Schema Parser sub-module is responsible for parsing the target XML Schema and retrieves any valuable information it's stored in its structure, e.g. annotations used for documenting the schema. After parsing the XML schema the sub-module generates an intermediate data structure serialized using the JSON language in order for the user interface to parse and generate the corresponding visual components. The rationale behind choosing JSON as the serialization language for that data structure is the software interoperability the Ingestion platform is attempting to achieve. JSON is a well supported language with interpreters for every major language platform available which reduces the overhead introduced by using XML for exchanging messages both by a reduced memory footprint needed and a simpler structure which makes parsing a much easier and lightweight task. The XML Schema Parser sub-module requests and retrieves the schema needed from the

persistent data layer. By using Hibernate for accessing and manipulating the data model, the software's architecture ensures the platform neutrality and separates the maintenance of the sub-modules from that of the data model itself.

The XML Schema parser sub-module is based on the XML Schema Object Model (XSOM) API that is part of the JAXB API for XML data binding. The main design goals of the XSOM API are a) to expose all the defined in the schema spec and b) to provide additional methods that help simplifying client applications. XSOM consists of roughly three parts; the first part is the public interface the entire functionality of XSOM is exposed by this interface to the client. The second part is the actual implementation of these interfaces. Finally the third part is a parser that reads XML representation of XML Schema and builds the XSOM data model accordingly. This part of the code is mainly generated by the RelaxNGCC API.

For the needs of the CARARE Ingestion service, an import is not required to include the schema used. This simplifies the actual work for the user and at the same time the set of schema components that have to be mapped is reduced to only those that are used, thus reducing redundancy. The Schema Generator sub-module produces the required simplified version of the schema that corresponds to a specific import by the user. When a user triggers the invocation of the mapping tool for a specific import, this sub-module is also invoked. It communicates with the data layer using the Hibernate persistent API. The next step in the workflow of the Schema Generator sub-module is to parse the data for a specific import and generate a tree like structure using HTML elements that represents the schema used. This tree like structure is then transmitted to the User Interface sub-module and is enhanced using JavaScript in order to create an interactive tree that represents a snapshot of the XML schema that the user is going to use as input for the mapping process.

The User Interface sub-module is responsible for creating and presenting an intuitive and visual appealing environment for the user to define mappings, without sacrificing any of the functionality needed to properly achieve the task of schema mapping. This sub-module is invoked by the user through the Overview interface of the Ingestion platform per import listed there. When the invocation occurs the server retrieves the id of the import and the workflow of the mapping tool is executed; the final step of that workflow is the transmission of all the appropriate structures to the user's browser where the mapping tool is rendered. The User Interface sub-module is implemented in



JavaScript using the YUI library from Yahoo. The usage of that library for implementing the visual components also ensures cross-browser compatibility.

6. Related Work

This chapter provides an overview of relevant platforms and tools that deal with ingesting, mapping and transforming metadata records as well as with enabling permanent access to digital works.

The HP-MIT DSpace Repository project

The DSpace project was initiated in July 2000 as part of the HP-MIT alliance. In 2007 the DSpace foundation was formed as a non-profit organization to provide support to the growing community of institutions that use DSpace. The foundation's mission is to lead the collaborative development of open source software to enable permanent access to digital works.

DSpace is a platform that allows you to capture items in any format – in text, video, audio and data in general with the purpose of distributing it over the web. It indexes the data so users can search and retrieve the items that constitute it. Moreover, another major functionality of DSpace is the ability to preserve the data over long term. It is typically used as an institutional repository supporting the following three roles:

- Facilitate **capture** and **ingest** of materials, including any related metadata.
- Facilitate **easy access** to the materials, both by **listing** and **searching**.
- Facilitate the **long term preservation** of the materials.

Finally, DSpace can be used to store any type of digital medium, e.g. videos, images, data sets, journal papers and others. The overall architecture of DSpace is presented in Figure 6.1.

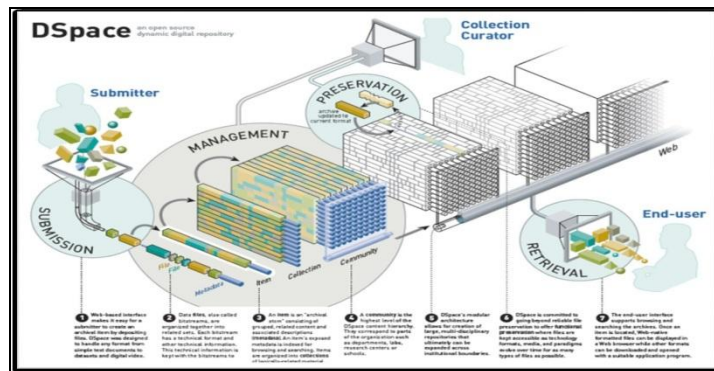


Figure 6.1 the over architecture of the DSpace platform.

DSpace is designed to work “out of the box” for basic repository needs while still being customizable; the system has been put to a wide variety of uses, and has been entrusted with important intellectual content produced by many institutions. While a certain amount of evolution can take place simply by patches, contributions and reimplementations of specific components the DSpace foundation recognized the necessity of a major review of the core architecture of DSpace, motivated mainly from the widespread adoption of the platform and the technological developments that occurred from the initial design of DSpace back in 2000. For this reason a group of experts and DSpace committers was formed in 2004 that would be responsible of re-evaluating the DSpace platform and propose a set of technical characteristics the DSpace Version 2.0 should have. The outcome of this group can be summarized in the following list of principles. These principles will govern the development of the next version of DSpace.

1. DSpace should be primarily open source software for building digital repositories.
2. DSpace should be usable based purely on free and open source software.
3. DSpace should have a decoupled, stable and application neutral core.
4. DSpace should be usable for a variety of applications but at the same time it will retain useful “out of the box” functionality for common use cases.
5. DSpace should employ and support existing, open standards where possible and practical.
6. DSpace releases should be minimal disruptive.
7. DSpace should support an exit strategy for content.
8. DSpace should evolve.

Based on these design principles the group of experts compiled a list of specific recommendations that would be part of the new version of DSpace. These recommendations attempt to tackle existing issues that appear in the current version of DSpace, e.g. scalability and interoperability among others.

The Fedora Digital Object Repository Management System

The Fedora digital object repository management system is based on the Flexible Extensible Digital Object and Repository Architecture (FEDORA). The system is designed to be a foundation architecture upon which full featured institutional repositories and other interoperable web based digital libraries can be built. It was jointly developed by the University of Virginia and Cornell University, the system implements the Fedora architecture, adding utilities that facilitate repository management. The current version of the software provides a repository that can handle one million objects efficiently. Subsequent versions of the software will add functionality important for institutional repository implementations, such as policy enforcement, and performance enhancement to support very large repositories. The system's interface comprises three web based services:

1. A management API that defines an interface for administering the repository, including operations necessary for clients to create and maintain digital objects;
2. An access API that facilitates the discovery and dissemination of objects in the repository;
and
3. A streamlined version of the access system implemented as an HTTP-enabled web service.

Fedora supports repositories that range in complexity from simple implementations that use the web service's "out of the box" defaults to highly customized and full featured distributed digital repositories.

Another characteristic of Fedora is that since it is a web service, it does not have a standard front end. Nevertheless, many UI applications have been implemented to front-end Fedora by the open source community that supports it.

One of the main strengths of the Fedora framework is that it demonstrates the best scalability among the most used repositories that exist. At the same time it easily supports the storage of multiple types of digital objects and collections particularly well. Another noticeable strength of the platform is that as a foundation architecture that provides powerful API based interoperability features, Fedora is highly flexible and powerful, and has proven itself with large networked repositories similar to those



envisaged with the OARINZ project. With no set user interface, Fedora has true separation between the ‘backend’ and ‘frontend’. Fedora provides good interoperability among different systems, with different options allowing for smart and flexible integration methods. Finally, it is supported by a strong development team and development map.

In a sense, a key strength can also be perceived as a weakness. With no user interface, Fedora cannot offer a full repository service ‘out of the box’ and therefore provides a conceptual complexity which other systems like DSpace do not. The code base of the Fedora platform is probably the largest among the commonly used repositories while at the same time the Fedora development community can be described as closed. These two weaknesses reduce the adoption of the Fedora platform by the repository community.

The EPrints repository platform

The EPrints software has probably the largest and most broadly distributed base of the majority of the repository platforms that exist. It was developed at the University of Southampton and the first version of the system was released in late 2000. The project is supported by JISC, as part of the Open Citation Project and by NSF. EPrints worldwide installed base affords an extensive support network for new implementations. The size of the installed base for EPrints suggests that any institution can get it up and running with minimal effort or technical expertise. Moreover, the number of EPrints installations that have augmented the system’s baseline capabilities, for example by integrating advanced search, extended metadata and other features, indicates that the system can be readily modified to meet local requirements.

As already mentioned, the EPrints platform is a good candidate as the repository platform of choice for many institutions because it is one of the least complex systems in existence and hence it has a low skill barrier to implement and maintain. At the same time, because it has one of the widest install bases, it goes a long way to ensure its longevity as a fully supported system. Finally, the code base of EPrints is uniform and well documented making it easier to work on for low level customization.

A major weakness of EPrints lies in the data model used which causes some scalability issues, although these could be addressed with some development effort. Also, its method of adding new digital content type can lead to disparate data models and compatibility issues if maintaining multiple



systems. Finally, the development team of EPrints denies any external contribution to the code base of EPrints.

The CERN Document Server Software (CDSware)

The CERN Document Server Software (CDSware) was developed to support the CERN Document Server. The software is maintained and made publicly available by CERN (the European Organization for Nuclear Research) and supports electronic preprint servers, online library catalogs and other web based document repository systems. CERN uses CDSware to manage over 350 collections of data, comprising over 500,000 bibliographic records and 220,000 full text documents, including preprints, journal articles, books and photographs.

CDSware was designed to accommodate the content submission, quality control, and dissemination requirements of multiple research units. Therefore, the system supports multiple workflow processes and multiple collections within a community. The service also includes customization features, including private and public baskets or folders and personalized email alerts.

CDSware was built to handle very large repositories holding disparate types of materials, including multimedia content catalogs, museum object descriptions and confidential and public sets of documents. Each release is tested live under the rigors of the CERN environment before being publicly released.

The CDSware exhibits the following major weaknesses,

- It has extremely complex installation steps.
- CDSware also does not have a good community around it. The mailing list has had very limited traffic since 2002, which indicates that this project may have sustainability issues going forward.

DRIVER: Building a sustainable infrastructure of (European) Scientific Repositories

The DRIVER platform which is the outcome of the European funded e-infrastructure project “DRIVER: Building a sustainable infrastructure of (European) Scientific Repositories”, does not

constitute a repository platform but a framework for creating and managing a network of existing repositories. The main aims and objectives of the Driver platform are the following:

- To organize and build a virtual, European scale network of existing institutional repositories.
- To assess and implement state-of-the-art technology, which manages the physically distributed repositories as one large scale virtual content resource.
- To assess and implement a number of fundamental user services
- To identify, implement and promote a relevant set of standards
- To prepare the future expansion and upgrade of the DR infrastructure across Europe and to ensure the widest possible involvement and exploitation by users.

Version 1.0 of the D-NET Software: Driver network-Evolution-Toolkit is already released under the Apache open source license to the public including the following modules:

- Repository network administration software (such as the Repository Network Manager, Resource Monitoring and others).
- End User services (search, browse, profiling).
- Support service to local repository managers and aggregators (Validation Tool).

The current Driver infrastructure supports three groups of users, A) the repository manager, B) the service provider and C) the researcher, reader, public. Apart from only providing the appropriate technological tools to support the creation and maintenance of a repository network, Driver also defines and supports the concept of the European Community of repository networks. “Community” means that the members agree to some fundamental principles and that the Driver community is wider than the Driver consortium and has no legal restrictions and is open to new members. Some of these fundamental principles are listed below:

1. Make research publications open to the public.
2. Become partner in a repository service network.
3. Follow “guidelines” to make data and services interoperable.
4. Ensure long term access to an institution’s research publications.

REPOX – A Metadata Space Manager

Repos is a framework to manage metadata spaces. It comprises several channels to import metadata from data providers, services to transform metadata between different schemas according to user's specified rules, and services to expose the results to the exterior. This tailored version of Repos aims to provide to the TEL partners a simple solution to import, convert and expose their bibliographic data via OAI-PMH, by the following means:

- Cross platform. Repos is developed in Java so it can be deployed in any operating system that has an available Java Virtual Machine.
- Easy deployment. Repos is available with an easy installer, which includes all the required software and libraries.
- Support for several metadata formats. Repos currently supports MARC21, UNIMARC, MarcXchange and MARCXML schemas out of the box and encodings in ISO 2709 (including several variants).
- Metadata crosswalks. It offers crosswalks for converting MARC21 and UNIMARC records to simple Dublin core as also to TEL-AP. A simple user interface makes it possible to customize these crosswalks and create new ones for other formats.

Repos is not a complete repository platform, although it imports metadata and stores them in a custom format for easy access providing at the same time a way of exposing these metadata to the web using an implementation of the OAI-PMH protocol for exchanging metadata over the web. It also includes a mapping tool capable of mapping various input metadata schemas to the TEL format. For this reason, in its current state Repos is limited to support only the exposure of metadata transformed in the format defined and supported by the TEL project.

7. Conclusions

The deliverable presented the web services that have been implemented in CARARE project in order to ingest content providers' metadata in CARARE repository. The web services that consist of the CARARE ingestion tool are the following, the harvesting-delivery, mapping, statistics, transformation and annotation. The relevant work, past and ongoing, has also been presented in this deliverable.



8. References

CARARE, 2010, Papatheodorou, C., Carlisle, P., Ertmann-Christiansen, C. and Fernie, K., CARARE metadata schema outline, v1.0: <http://www.carare.eu/eng/Resources/CARARE-metadata-schema-outline-v1.0>

Europeana, 2010, Europeana Data Model Definition v.5.2

NTUA, 2010, Metadata Interoperability Services: <http://mint.image.ntua.gr>

W3C, 1999, XSL Transformations: <http://www.w3.org/TR/xslt>

W3C, 1998, XML: <http://www.w3.org/TR/xml>

Repos, <http://repos.ist.utl.pt/>

Fedora, <http://fedoraproject.org/>

Dspace, <http://www.dspace.org/>

Driver, <http://www.driver-repository.eu/>