ECP-2007-DILI-517005

ATHENA

First version of the semantic interoperability plug-in

Deliverable number	*D7.1*
Dissemination level	*Public*
Delivery date	*30 November 2009*
Status	*Draft*
Authors	*Vassilis Tzouvaras, Nasos Drosopoulos, Kostas Pardalis, Anna Christaki, Arne Stabenau, Fotis Xenikoudakis and Stefanos Kollias – NTUA (GR)*

e*Content*plus

This project is funded under the eContentplus programme[1],
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

Table of Contents

1 Executive summary

Present document constitutes the first report for the design and development of the web-based tool that will be used for the ingestion and publishing of cultural content metadata belonging to providers that participate in the ATHENA project. This process aims to guarantee semantic interoperability across numerous data repositories of varied thematic categories, technical features and capabilities, allowing robust ingestion of diverse content and knowledge. The results of requirements analysis together with the outcome of close cooperation with relevant work packages and working groups of the project are illustrated and, the subsequent decision making process that has lead to the final functional and technical requirements of the prototype are discussed. Additionally, the core modules that currently serve the ingestion process are described, together with examples and screens that cover all the functionality of the implemented web service. This document can be used as a reference for the prototype design and implementation as well as for providing instructions for the service's usage, modules and features.

The tool is developed based on a platform implemented and maintained by the WP leader - IVM Laboratory of the National Technical University of Athens - and is customised and extended to support the consortium needs through close cooperation with all relevant work packages.
The basic version of the mapping tool will be released under an open-source license. This decision has been taken also due to the Europeana's intentions to reusing it. The whole application includes other software parts that have been sometimes tailored for the specific purposes of the ATHENA project.

Primary focus is on supporting aggregation of arbitrary provider organisation data models, adopting the Lightweight Information Describing Objects (LIDO v0.7) as the reference metadata schema and, publishing aggregated content in the Europeana Semantic Elements schema. The basic steps in the process that leads to semantic interoperability and allows for the ingestion and aggregation of all cultural heritage content within ATHENA and the subsequent publishing of the semantically interoperable metadata, specifically for harvesting by the Europeana portal, are as follows:

- registration and access rights for users and their respective organisations, supporting national or thematic 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 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; focus of deployment is interoperability with Europeana.

The application will be hosted by the server of NTUA.

2 Introduction

WP7 is responsible for the semantic alignment of metadata schemata that content providers use to annotate their items, to a common, well-defined, machine understandable schema, which in turn will allow for the ingestion of aggregated content in the Europeana portal. Content providers currently participating in the ATHENA project, as well as several that are contemplating their contribution during the project's life cycle, constitute mainly of museums and in a lesser extent libraries and archives. As it was illustrated through WP3, and specifically deliverable D3.1 'Report on existing standards applied by European museums', as well as from the various user requirements and analysis studies and meetings that took place within the first year of the project, there is a large variety of metadata standards used throughout the providers. There are a limited number of key standards, and they are used extensively throughout Europe and indeed the world. These are often suggested as best practice but, from the evidence of the WP3 survey, there is still a long way to go to achieve interoperability. National standards in some countries are also a factor that needs to be taken into consideration. There is also often the case where providers bypass the semantics of a standard through proprietary rules and conventions that are not sufficiently documented or semantically defined, resulting in misinterpretations that can not be straightforwardly detected. Finally, there is a vast variety, with respect to the level of detail, in annotation, ranging from the use of a small flat-structured set of elements to defining and using complex schemata that support various levels of annotation, item and concept relations and connections with controlled vocabularies and thesauri.

In this context, joint efforts from WP3 and WP7 have identified the need for an intermediate standard in ATHENA that will serve as the point of interoperability between the providers, as well as between the project's repository and outside sources. These efforts led in the adoption of LIDO (Lightweight Information Describing Objects) v0.7, an XML Schema for contributing content to cultural heritage repositories. LIDO is the result of a joint effort of the CDWA-Lite, museumdat and Spectrum communities and meets requirements for publishing metadata from all kinds of cultural object classes. It integrates the relevant concepts of the aforementioned schemata into one single schema addressing several important issues that include an event-oriented approach, refined subject element, full support of multilingualism, revised handling of display and index elements and, revised identifier handling (for semantic web needs). In the light of recent and future Europeana developments, LIDO also allows for the contribution of metadata along with digital representations of items, delivers information in a 'self-contained' way, includes links to original repository, distinguishes between display and indexing elements and provides references to controlled vocabularies and authorities.

The adoption of LIDO, coupled with the loosely defined providers' input schemata, led WP7 to design an aggregation and mapping workflow that will allow 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 will apply to the actual metadata records, which subsequently (re-)define the input schema in a semantic, machine understandable way, based on its mapping to LIDO.

The web service aims to provide a user friendly ingestion environment that allows for the extraction and presentation of all relevant and statistical information concerning input metadata together with an intuitive mapping service that illustrates LIDO 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 using respective transformation functions (e.g. concatenate) that will allow for the addition and enrichment of semantics even when those are not specifically stated in the input.

In this context, the tool also determines the operational work flow processes to bring the amalgamated content of the partner museums into Europeana and to create, manage and execute, with the European Digital Library Office, the implementation plan to ensure that the content is visible in Europeana. The platform supports exporting of the aggregated metadata to several established standards concerning presentation and archive management. Primary effort is directed to the transformation of the aggregated content to the Europeana Semantic Elements schema and the deployment of an OAI-PMH repository to facilitate harvesting by Europeana.

The ingestion tool is based on a metadata aggregation platform that is developed and maintained by the leader of the WP, IVM Laboratory of the National Technical University of Athens, and is designed based on a specialisation of a general metadata ingestion work-flow, as it is described in detail in Chapter 3.

The rest of this document is structured as follows:

Chapter 3 discusses the Ingestion Analysis and the Workflow that was designed for the ATHENA project, illustrating the processes a cultural heritage provider undertakes to enable the presentation of its content through the Europeana portal. Chapter 4 documents the Harvesting and Delivery 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 5 outlines the Mapping module as it is set up to support LIDO as an intermediate schema, together with relevant functionalities that enable semantic interoperability for the ingested content. Chapter 6 gives an overview of the Statistics module and relevant information and functionality that it presents to the user guiding the mapping process. Chapter 7 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, Chapters 8 discusses future developments towards the final prototype of the project's tool while Chapter 9 summarizes the Conclusions of the report.

3 Ingestion analysis and workflow

The aim of the ATHENA project is to serve as a vertical metadata aggregator (i.e. Museums metadata) for Europeana. The aim of WP7 is to provide the technical facilities to accomplish this task.

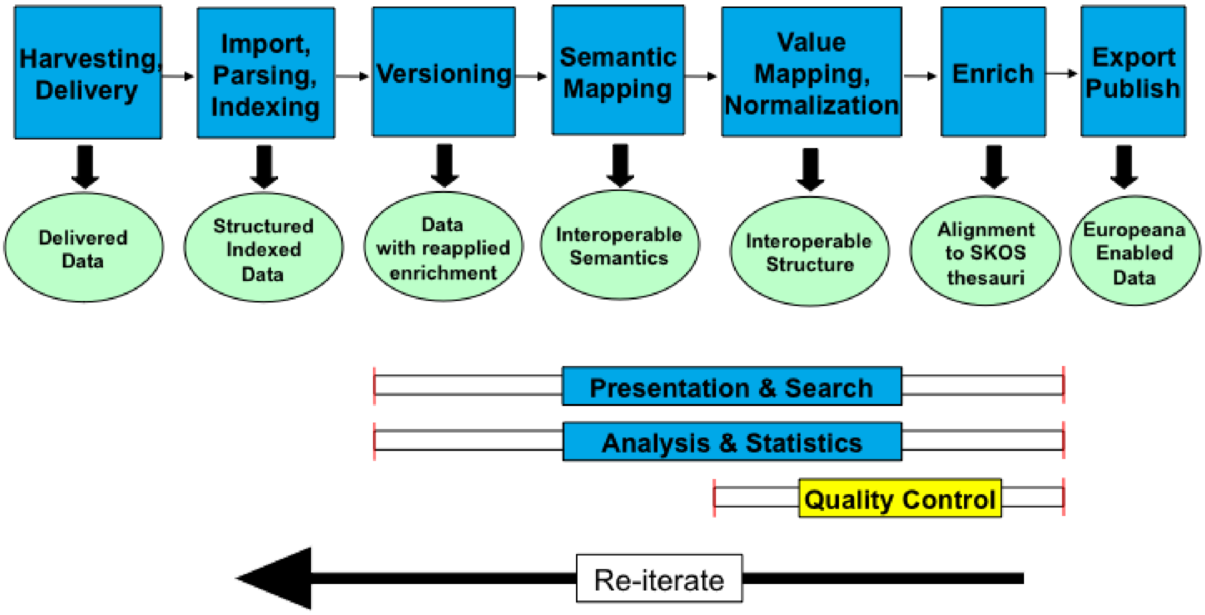


Figure 4.1 Ingestion workflow

In the first phase of ingestion for Europeana, a number of problems were noted that a) delayed the process b) created a lot of manual work due to the low quality of the ingested metadata and c) created a non flexible ingestion mechanism. As a result, a lot of man power was required due to the lack of automation and also, the providers could not update or correct the ingested metadata, ending up in some cases presenting incorrect data. The aim of WP7 is to provide an infrastructure capable to ingest quality metadata to Europeana as well as to provide automation and flexibility. In Figure 4.1 we present the ingestion workflow of the ATHENA project.
The workflow consists of seven phases. Each phase is responsible for specific services needed to ensure the quality of the ingestion process.

Harvesting-delivery
It is responsible to set up the environment within which the system will harvest the metadata from the repositories placed at provider’s site. It is not using a predefined harvesting format at this phase since there is need for harvesting, indexing and storage of all the available metadata from the content providers. It is an interface for different methods of data delivery including:

- OAI-PMH
- HTTP upload/download
- FTP upload/download

The data format is XML. The content providers will have the ability to declare the upload:

- complete (new or updated or deletes)
- partial (new or updated or mixed)

Import, Parsing and Indexing

It allows for unified access to data. With this tool all Content Provider data is structurally accessible the same way, e.g. xpath-like access for subsequent processing.

Versioning

It is responsible for the management of uploads and the formation of the provider's input repository with respect to incremental or overlapping uploads. Sophisticated comparison algorithms that detect duplicate or edited items are applied for upload dataset comparison. This functionality allows for future development of services like:

- Ability to transfer edits & enrichments on old versions of items to the current repository.
- Merging uploads that require overwriting of existing items that were not edited or enriched.
- Conflict resolution in cases where multiple edits occur for a specific item.

Semantic Mapping

It provides the service for assigning semantics to the harvested raw metadata. It is used to manually map providers' fields to LIDO schema. The mapping transformations are stored and the provider is able to edit them at any point if necessary. It supports value concatenation (many-to-one mappings) and conditions with one-to-many mappings. Providers that have metadata in supported known formats might be able to omit this step (use stored transformations from selected schemas to LIDO based on existing crosswalks).

The software includes examples provided by the LIDO team and by the WP3 workgroup, as well as the outcomes of the demo workshops and training material that will help providers to understanding how their own schema is able to be adapted to the complexity of the process. In any case, mapping is a task to be carried out by the content providers, since it must support the transfer of knowledge about the semantics of their own schemas, that they only know can describe. The support provided by from ATHENA project consists of providing and facilitating the use of the ingestion tool providing a user friendly way to visualize mappings, avoiding the difficulties of a completely manual process run by the content providers to write down firstly their own mappings and then a programme to perform the actual transformations.

Value Mapping, Normalization

It allows for the alignment of controlled vocabularies between the input and target schema and for the normalization of element values according to required conventions. In particular:

- It enables providers to resolve data issues, e.g. map own terminology list to selected terminology lists.
- It automatically normalizes data e.g. dates, geographical locations, nationality/language, name writing convention to Europeana standards.

Enrichment

It enables the addition of data that is not in the original metadata (e.g. empty fields, fields that take values corresponding to identifiers for controlled vocabularies). This service will also enable the use of enrichment to add values from the SKOS thesauri. The enrichment will happen on item or group level.

Export-Publish

It creates the OAI-PMH repository with Europeana enabled data. This tool exposes metadata in the ESE schema for the time being and in richer formats in the future.

One of the reasons of using a reference model is to have the opportunity to re-export the repository to any new Europeana or external model by mapping only LIDO to it, instead of remapping all the content providers separately. In any case, the EDM is not focused on the representation of the actual metadata but more on the relations you can define between different annotations or between concepts that appear in the metadata (actors, locations etc).

Presentation & Search

It is responsible for previewing the metadata in the Europeana test environment. For this reason, the Sand-Box tool developed by Europeana has been used to test the correctness and the quality of the metadata.

Analysis & Statistics

This service provides detailed analysis and statistics of metadata contributed by a provider (i.e. number of items imported, total values per field etc).

Quality Control

This service will automatically check and report on Content Provider’s data (i.e. missing values, malformed data). Error reports and warnings from the tool can be used to edit semantic mappings, value mappings and/or edit items until the Provider’s data successfully passes the Quality control checks.

4 Harvesting and delivery service

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.

Content upload

Currently, allowed data formats for uploads are:

- XML in any schema.
- Excel files, based on a template
- zip archives of the above

Uploading of thumbnails or really any binary data will be supported in the future.

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 can not access data non related to that. 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

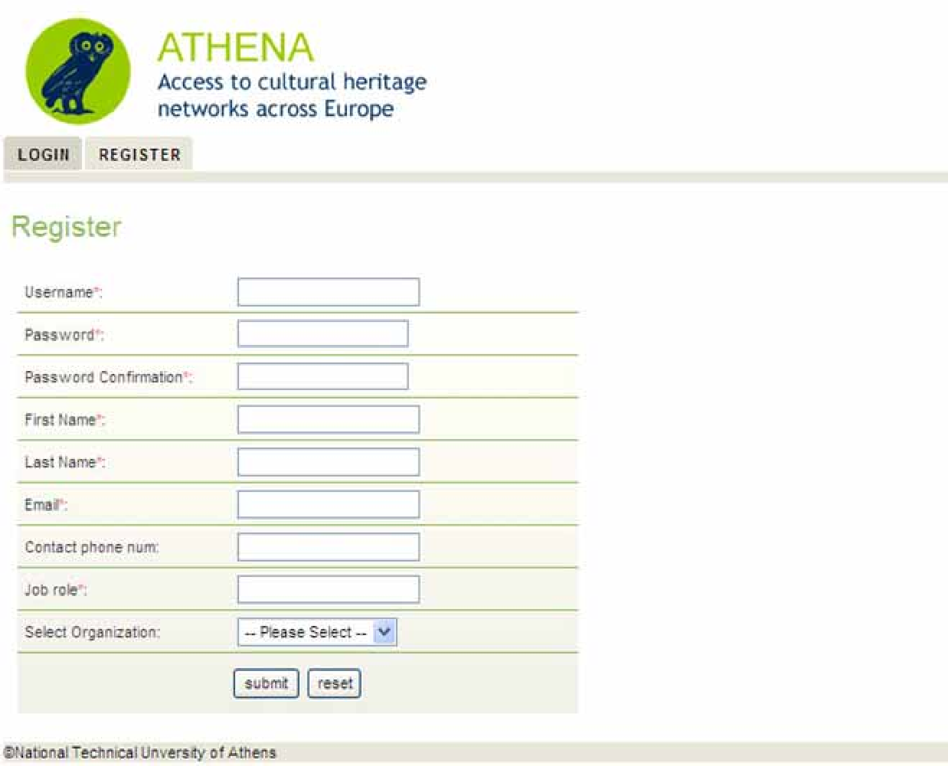
- edit/upload/delete data
- publish / declare finished datasets
- read-only browsing rights

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 ATHENA service using the registration page (fig. 4.1). During registration they are prompted to select the organization they belong to.



Register

Username*:

Password*:

Password Confirmation*:

First Name*:

Last Name*:

Email*:

Contact phone num:

Job role*:

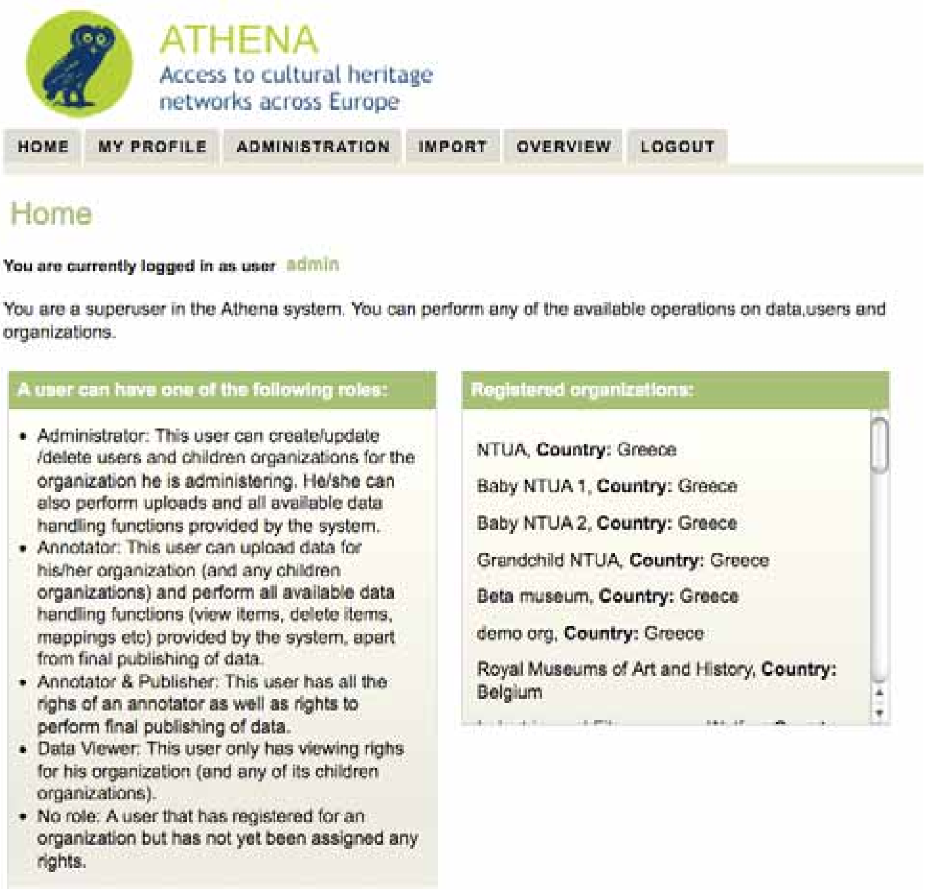
Select Organization: -- Please Select --

©National Technical Unversity of Athens

Figure 4.1 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

You are currently logged in as user admin

You are a superuser in the Athena system. You can perform any of the available operations on data,users and organizations.

A user can have one of the following 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 righs of an annotator as well as rights to perform final publishing of data.
- Data Viewer: This user only has viewing righs 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:

NTUA, **Country:** Greece

Baby NTUA 1, **Country:** Greece

Baby NTUA 2, **Country:** Greece

Grandchild NTUA, **Country:** Greece

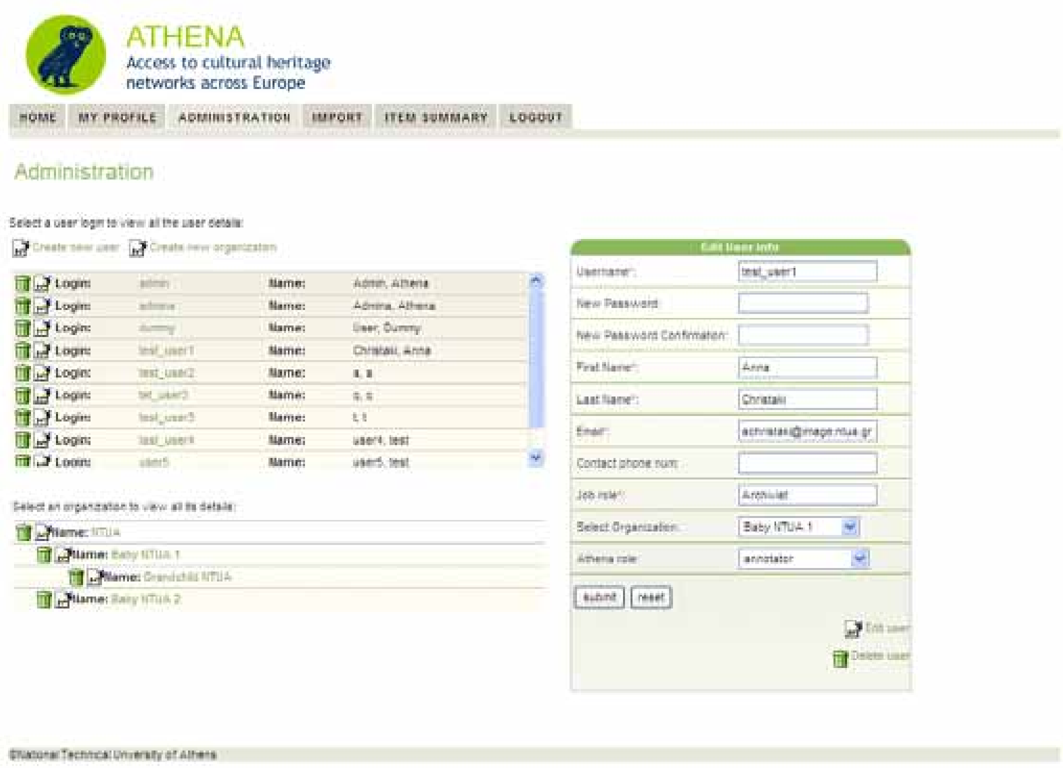
Beta museum, **Country:** Greece

demo org, **Country:** Greece

Royal Museums of Art and History, **Country:** Belgium

Figure 4.2 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 ATHENA.



Administration

Select a user login to view all the user details:

Login:	admin	Name:	Admin, Athena
Login:	admina	Name:	Admina, Athena
Login:	dummy	Name:	User, Dummy
Login:	test_user1	Name:	Christaki, Anna
Login:	test_user2	Name:	s, s
Login:	tet_user3	Name:	s, s
Login:	test_user3	Name:	t, t
Login:	test_user4	Name:	user4, test
Login:	user5	Name:	user5, test

Select an organization to view all its details:

- Name: NTUA
  - Name: Baby NTUA 1
    - Name: Grandchild NTUA
  - Name: Baby NTUA 2

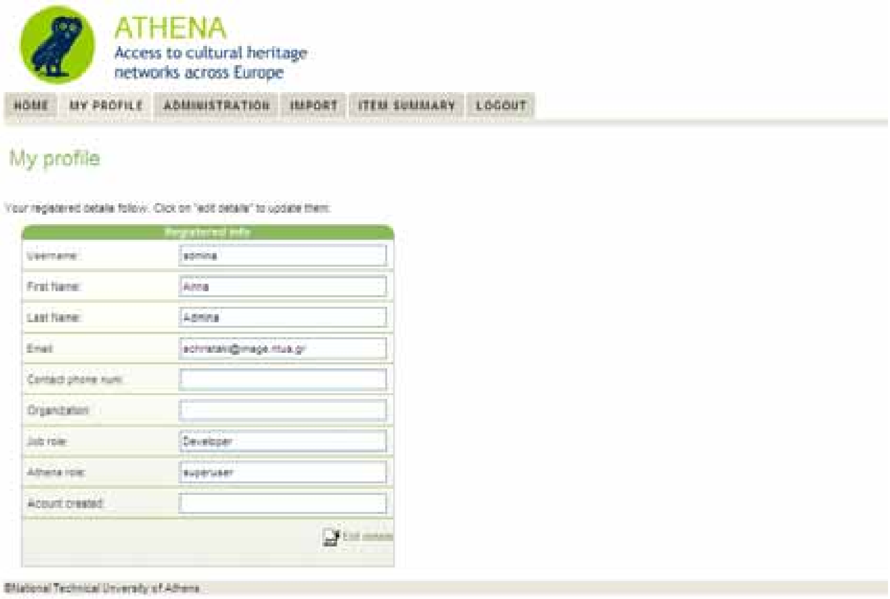
Edit User Info

Username*:	test_user1
New Password:	
New Password Confirmation:	
First Name*:	Anna
Last Name*:	Christaki
Email*:	achristaki@image.ntua.gr
Contact phone num:	
Job role*:	Archivist
Select Organization:	Baby NTUA 1
Athena role:	annotator

©National Technical University of Athens

Figure 4.3 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. 4.3). S/he can change her own details by using the Profile page (fig. 4.4).



My profile

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

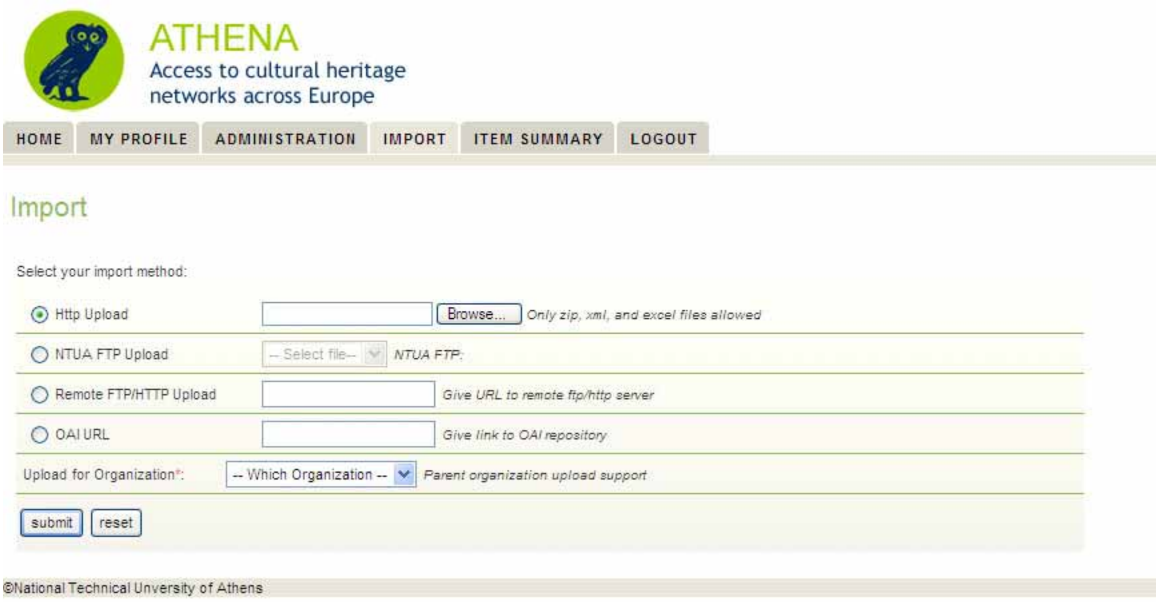
Registered info

Username:	admina
First Name:	Anna
Last Name:	Admina
Email:	achristaki@image.ntua.gr
Contact phone num:	
Organization:	
Job role:	Developer
Athena role:	superuser
Acount created:	

©National Technical University of Athens

Figure 4.4 User Profile Screen

Administrators and Annotators of an organization can upload data using the import page (fig. 4.5).



Import

Select your import method:

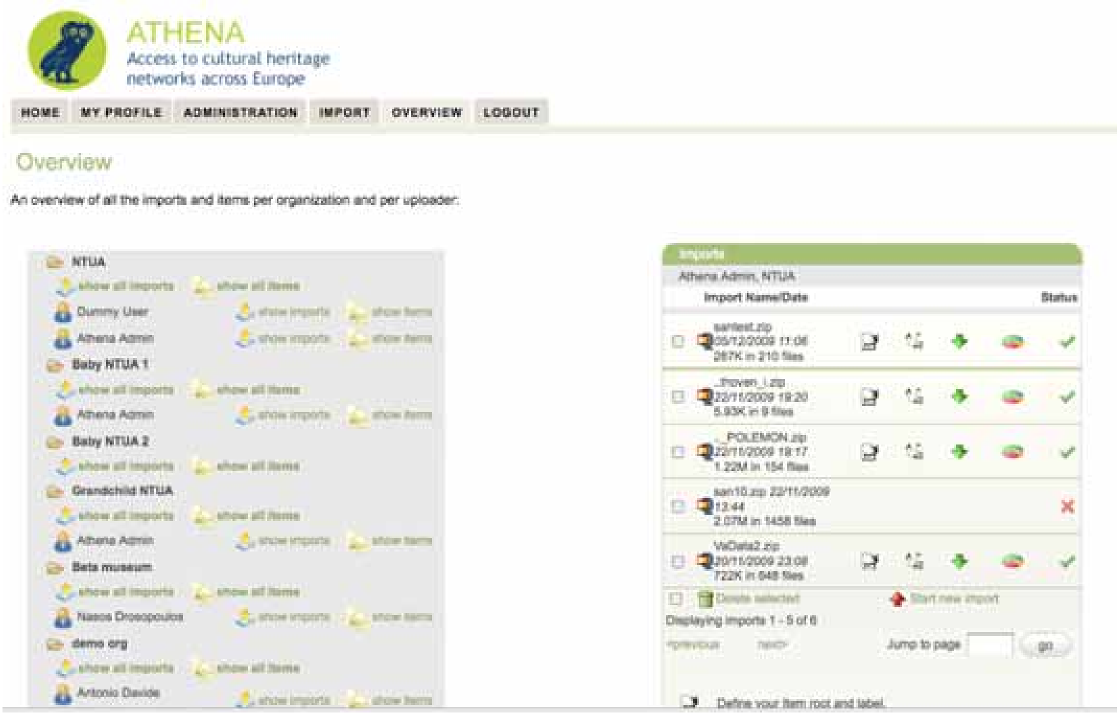
● Http Upload		Only zip, xml, and excel files allowed
○ NTUA FTP Upload	-- Select file--	NTUA FTP:
○ Remote FTP/HTTP Upload		Give URL to remote ftp/http server
○ OAI URL		Give link to OAI repository
Upload for Organization*:	-- Which Organization --	Parent organization upload support

©National Technical Unversity of Athens

Figure 4.5 Import Interface

A history of all uploads for an organization can be browsed using the Overview interface (fig. 4.6). 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.



Overview

An overview of all the imports and items per organization and per uploader:

NTUA
show all imports show all items
Dummy User show imports show items
Athena Admin show imports show items
Baby NTUA 1
show all imports show all items
Athena Admin show imports show items
Baby NTUA 2
show all imports show all items
Grandchild NTUA
show all imports show all items
Athena Admin show imports show items
Beta museum
show all imports show all items
Nasos Drosopoulos show imports show items
demo org
show all imports show all items
Antonio Davide show imports show items

Athena Admin, NTUA

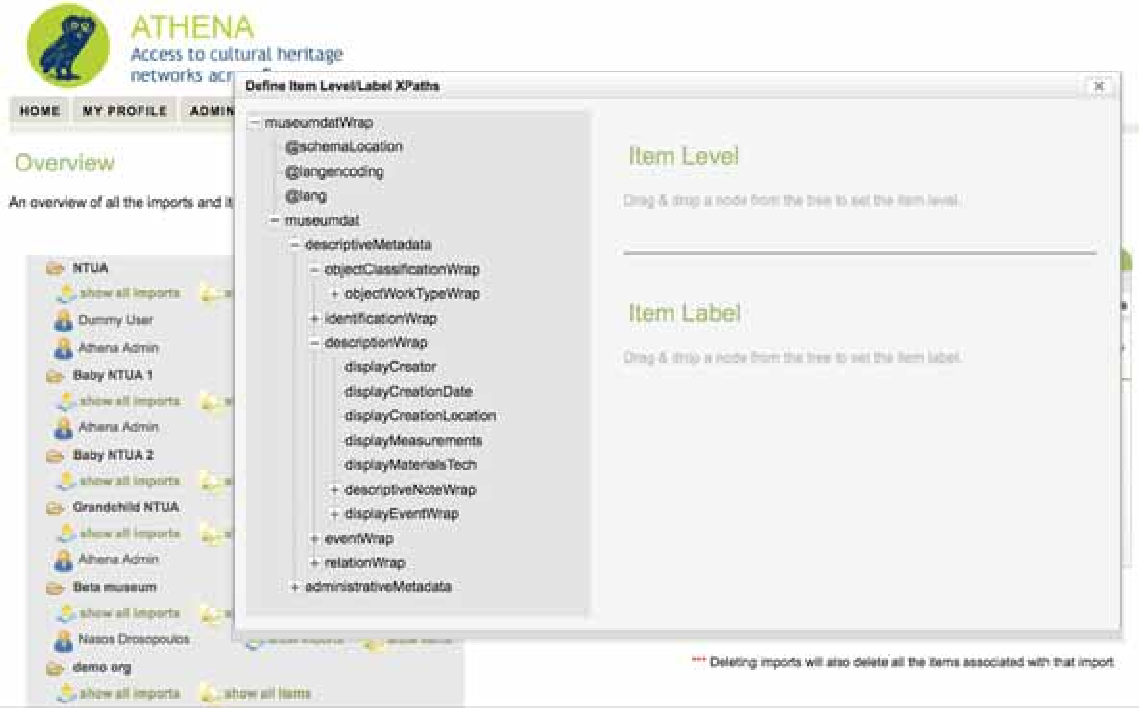
	Import Name/Date	Status
□	santest.zip 05/12/2009 11:06 287K in 210 files	✓
□	_thoven_i.zip 22/11/2009 19:20 5.93K in 9 files	✓
□	_POLEMON.zip 22/11/2009 19:17 1.22M in 154 files	✓
□	san10.zip 22/11/2009 13:44 2.07M in 1458 files	✗
□	VaData2.zip 20/11/2009 23:08 722K in 948 files	✓

Displaying imports 1 - 5 of 6

Define your item root and label.

Figure 4.6 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. 4.7) and additionally an element that will serve to label the separated items.



An overview of all the imports and it

Define Item Level/Label XPaths

```
− museumdatWrap
    @schemaLocation
    @langencoding
    @lang
  − museumdat
    − descriptiveMetadata
      − objectClassificationWrap
        + objectWorkTypeWrap
      + identificationWrap
      − descriptionWrap
          displayCreator
          displayCreationDate
          displayCreationLocation
          displayMeasurements
          displayMaterialsTech
        + descriptiveNoteWrap
        + displayEventWrap
      + eventWrap
      + relationWrap
    + administrativeMetadata
```

Item Level

Drag & drop a node from the tree to set the item level.

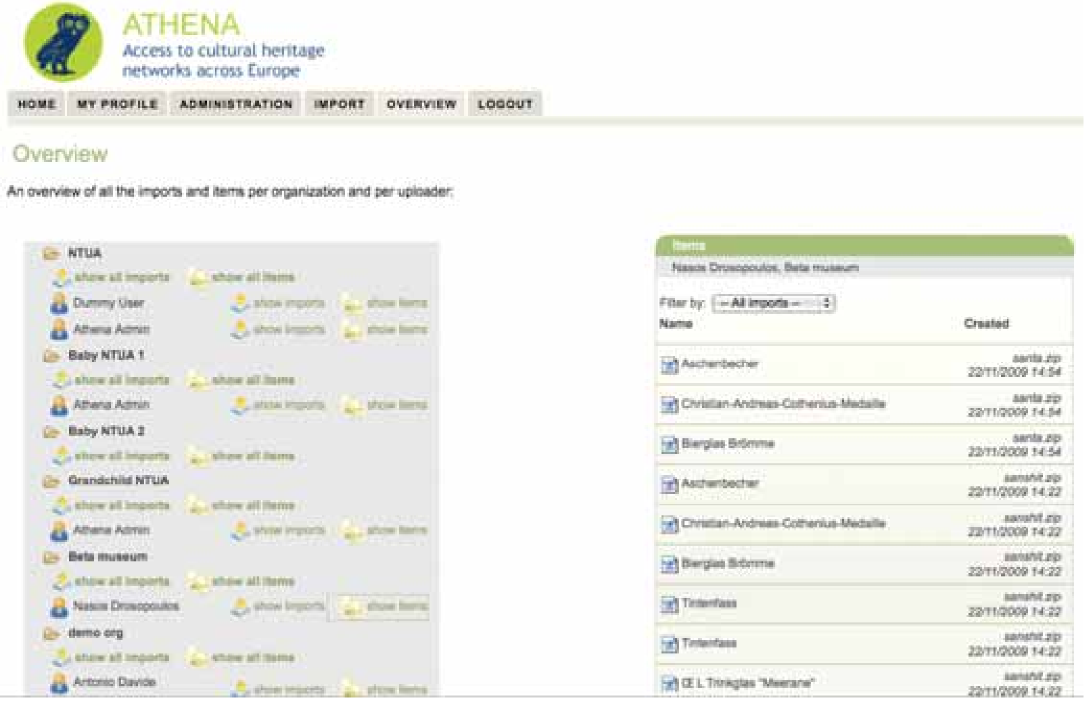
Item Label

Drag & drop a node from the tree to set the item label.

*** Deleting imports will also delete all the items associated with that import

Figure 4.7 Definition of item root element

Having defined the item root element, the items that belong to the specific upload can now be previewed (fig. 4.8). Finally, through an icon next to each item, the user views the original XML, the XSLT that has been produced through the mapping module (see Section 5), the XML output in the LIDO schema and, the XML output in the ESE schema (fig 4.9)



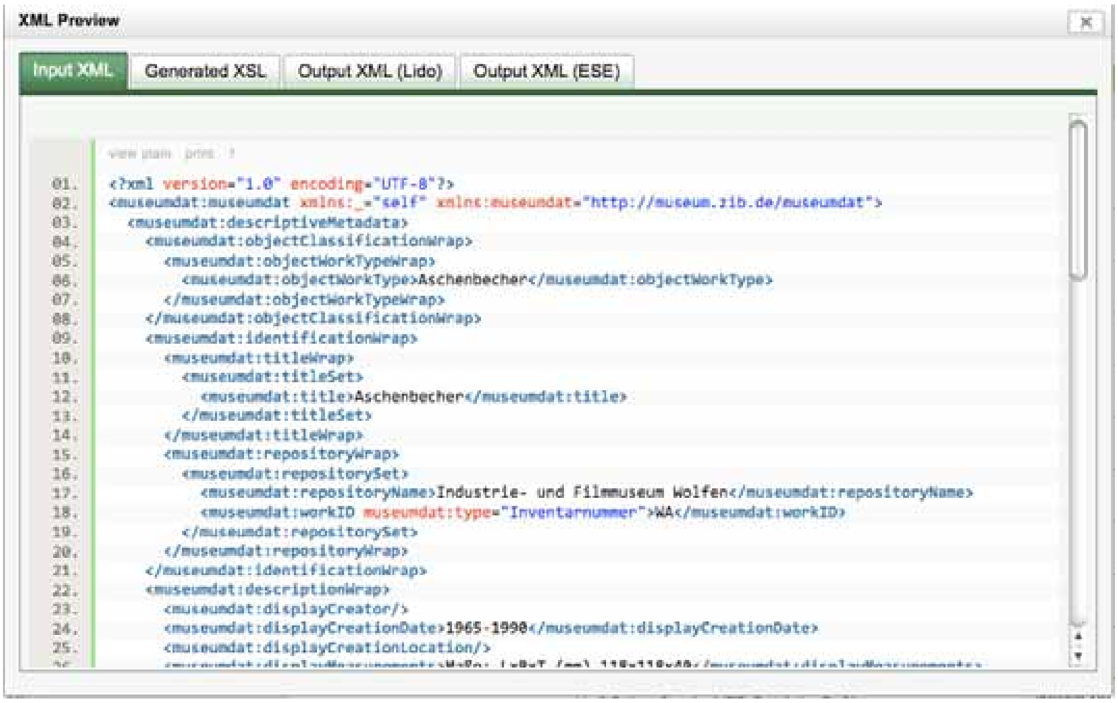
An overview of all the imports and items per organization and per uploader:

Nasos Drosopoulos, Beta museum

Filter by: -- All imports --

Name	Created
Aschenbecher	santa.zip 22/11/2009 14:54
Christian-Andreas-Cothenius-Medaille	santa.zip 22/11/2009 14:54
Bierglas Brömme	santa.zip 22/11/2009 14:54
Aschenbecher	sanshit.zip 22/11/2009 14:22
Christian-Andreas-Cothenius-Medaille	sanshit.zip 22/11/2009 14:22
Bierglas Brömme	sanshit.zip 22/11/2009 14:22
Tintenfass	sanshit.zip 22/11/2009 14:22
Tintenfass	sanshit.zip 22/11/2009 14:22
Œ L Trinkglas "Meerane"	sanshit.zip 22/11/2009 14:22

Figure 4.8 Items Screen



```
<?xml version="1.0" encoding="UTF-8"?>
<museumdat:museumdat xmlns:_="self" xmlns:museumdat="http://museum.zib.de/museumdat">
  <museumdat:descriptiveMetadata>
    <museumdat:objectClassificationWrap>
      <museumdat:objectWorkTypeWrap>
        <museumdat:objectWorkType>Aschenbecher</museumdat:objectWorkType>
      </museumdat:objectWorkTypeWrap>
    </museumdat:objectClassificationWrap>
    <museumdat:identificationWrap>
      <museumdat:titleWrap>
        <museumdat:titleSet>
          <museumdat:title>Aschenbecher</museumdat:title>
        </museumdat:titleSet>
      </museumdat:titleWrap>
      <museumdat:repositoryWrap>
        <museumdat:repositorySet>
          <museumdat:repositoryName>Industrie- und Filmmuseum Wolfen</museumdat:repositoryName>
          <museumdat:workID museumdat:type="Inventarnummer">WA</museumdat:workID>
        </museumdat:repositorySet>
      </museumdat:repositoryWrap>
    </museumdat:identificationWrap>
    <museumdat:descriptionWrap>
      <museumdat:displayCreator/>
      <museumdat:displayCreationDate>1965-1990</museumdat:displayCreationDate>
      <museumdat:displayCreationLocation/>
```

Figure 4.9 Input and output XML, and XSLT preview

5 Mapping module

Implementation details

The core module of the ATHENA web service is the mapping tool developed by NTUA. Although the service shares functionality with many existing metadata repositories, e.g. DSpace and Fedora, one of its main goals it to provide support for a great diversity of metadata schemas or simple data structures, thus widening metadata interoperability. The ATHENA 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 ATHENA 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 schema adopted in the ATHENA project is Lightweight Information Describing Objects (LIDO v0.7) which is expressive enough to accommodate the project’s content. 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. It can be extended to support alternative target schemas that are represented by valid XML documents.

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 ATHENA 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 various sub-modules of the mapping tool, together with their position in the overall architecture are presented in Figure 5.1.

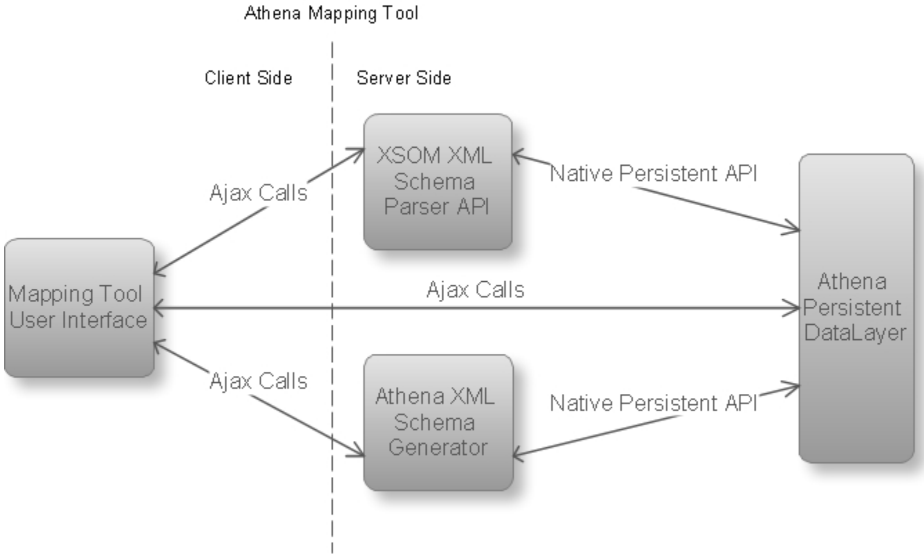


Figure 5.1 Architecture of the mapping tool.

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 ATHENA 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. The XML schema component model that XSOM is able to parse and represent is presented in Figure 5.2.

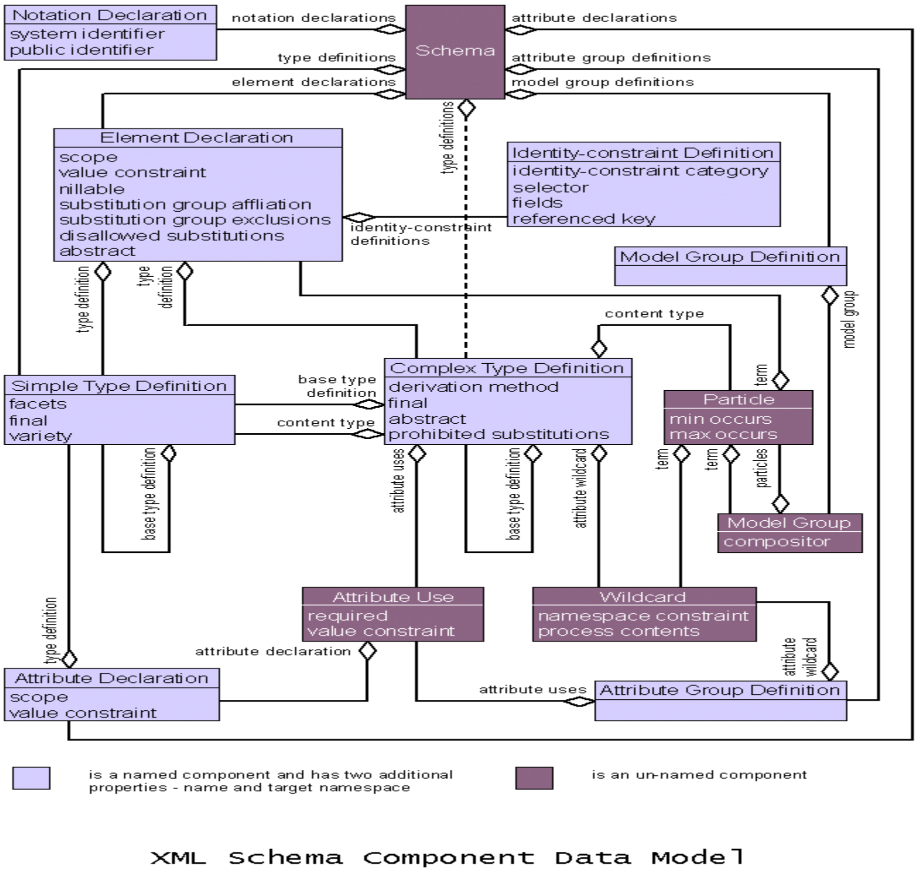


Figure 5.2 The XML Schema Component Data Model used with XSOM

For the needs of the ATHENA service, an import is not required to include the schema used.

This means that parsing and mapping are not based on standards that providers use, but on the actual structure of their metadata, regardless which standards it was based on. In this way, possible misuses of standards (which often happen) will not prevent content providers from being able to map their own content. Also any proprietary schema that providers use, can be mapped as well. If a provider is sure to using a standard correctly, e.g. museumdat, he/she can use an existing mapping to LIDO (created e.g. by the museumdat community) and transform these data without starting the mapping from the scratch.

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 ATHENA 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.

Functionality and user interfaces

In order to offer a more user friendly environment to perform the task of schema mapping, the tool can be configured to provide to the user groups of high level elements that constitute separate semantic entities. These top level sets of elements are presented on the right side of the mapping tool User Interface sub-module as can be seen in Figure 5.3. On the left side of the mapping tool User Interface a tree structure is always present that represents the schema produced by the Schema Generation sub-module for a specific import. The user is able to interact with this tree, expand or collapse the elements of the tree and retrieve brief statistics for each element and its values. The information for each element is retrieved asynchronously using AJAX calls from the server every time the user makes a request, in order to reduce the memory footprint on the client and the processing time on the server. An example of the info provided for each element can be found in Figure 5.4.

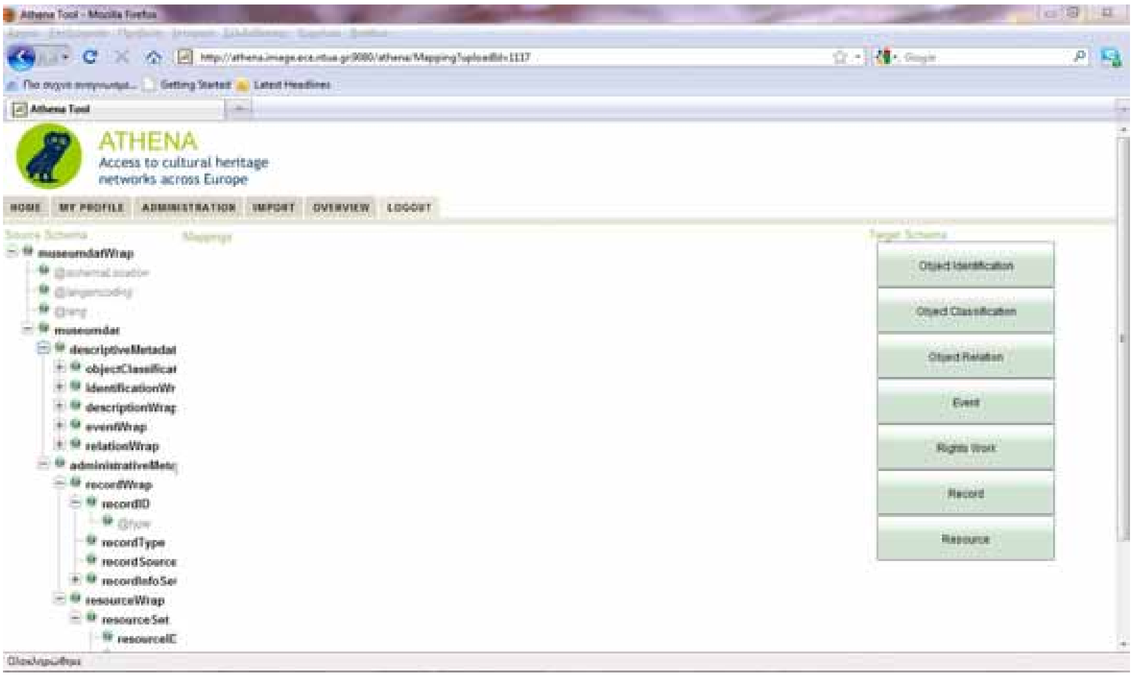
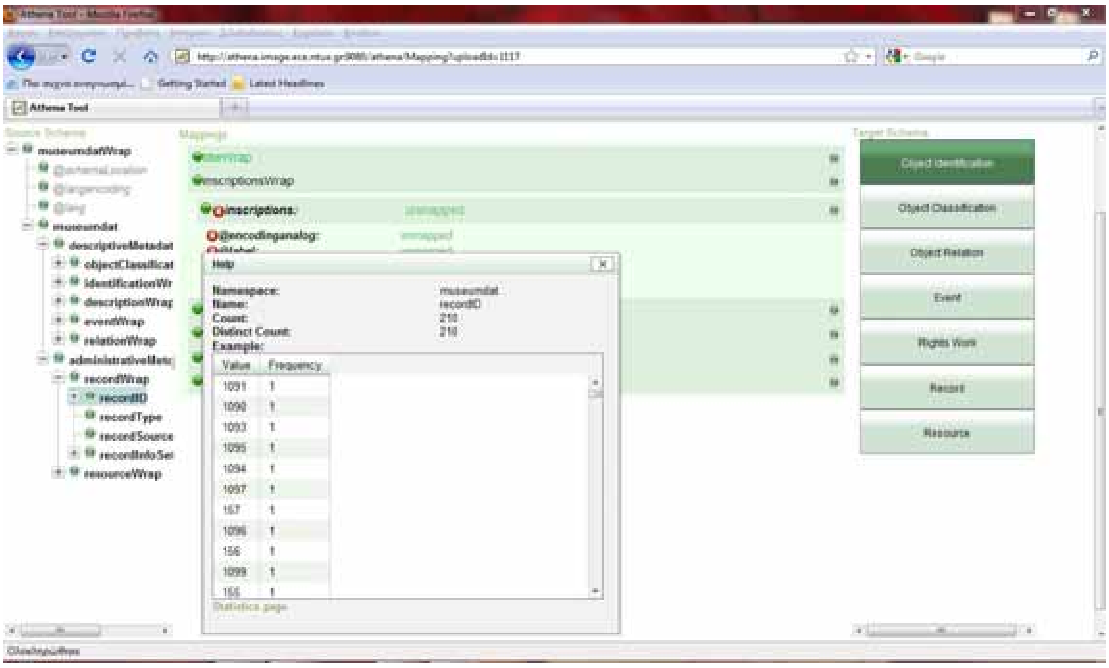


Figure 5.3 Screenshot of the mapping tool.



Namespace: museumdat
Name: recordID
Count: 210
Distinct Count: 210
Example:

Value	Frequency
1091	1
1090	1
1093	1
1095	1
1094	1
1097	1
157	1
1096	1
156	1
1099	1
155	1

Figure 5.4 Statistics for an input element

When a user wants to create or edit a mapping, he initially has to select one of the top level element groups that are presented on the right side of the mapping interface. Clicking the corresponding button, the set of the sub-elements that are part of that group are presented to him in the middle part of the screen. This part of the user interface has a tree structure of embedded boxes that represents the internal structure of the complex element. The user is able to interact with this structure by clicking to collapse and expand it, similar to what he is able to do the with the tree representation of the input schema. Every embedded box represents an element and the user is able to request and view any information about it that is part of the XML schema. This is achieved by pressing the information button on each element, which triggers an asynchronous request to the server where the XML Schema Parser sub-module returns any annotation that can be found for this particular element and the UI renders it on the screen using tooltips.

When a user wants to perform an actual mapping between the input and the target schema, he has to drag and drop any element he wishes from the tree structure on the left part of the user interface to one of the boxes in the middle. When a successful mapping occurs, the user gets notified for the event and he is able to view the mappings in the middle part of the screen. Using the delete button the user is able to delete and correct any mappings he has made so far and repeat the procedure.
The user interface of the mapping tool is completely schema aware regarding the target schema. That means that many operations might be restricted based on constraints that appear in the target XML schema. For example, if an element can be repeated the user is able by using a button that appears on the visual representation of that element to add another one and make a new mapping. Current development of the tool allows it to incorporate a set of functions in order to also ensure the datatype compatibility of the two schemas and provide to the user the ability to also transform the data apart of the schema itself.

The mappings are stored on the client using a data structure defined using the JSON language. Every time an event occurs that alters the current mappings this structure is also altered. The sub-module communicates directly with the persistent data layer and updates the stored structure of the mappings using asynchronous AJAX calls as can be seen in Figure 5.1. The usage of Hibernate for the data layer ensures the concurrency of the whole procedure in the case where many users might edit the same mappings for the same import simultaneously. A diagram depicting the various messages exchanged between the different sub-modules of the mapping tool and the rest of the system is presented in Figure 5.5.

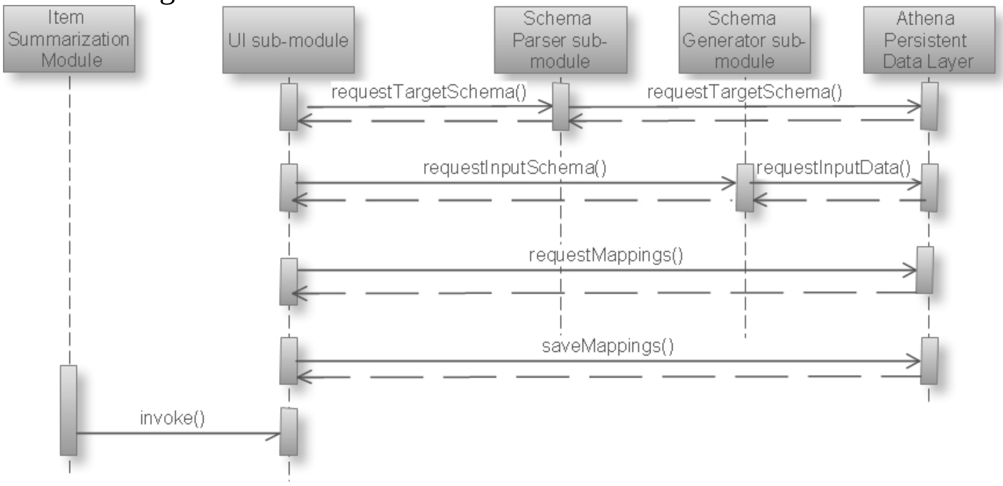
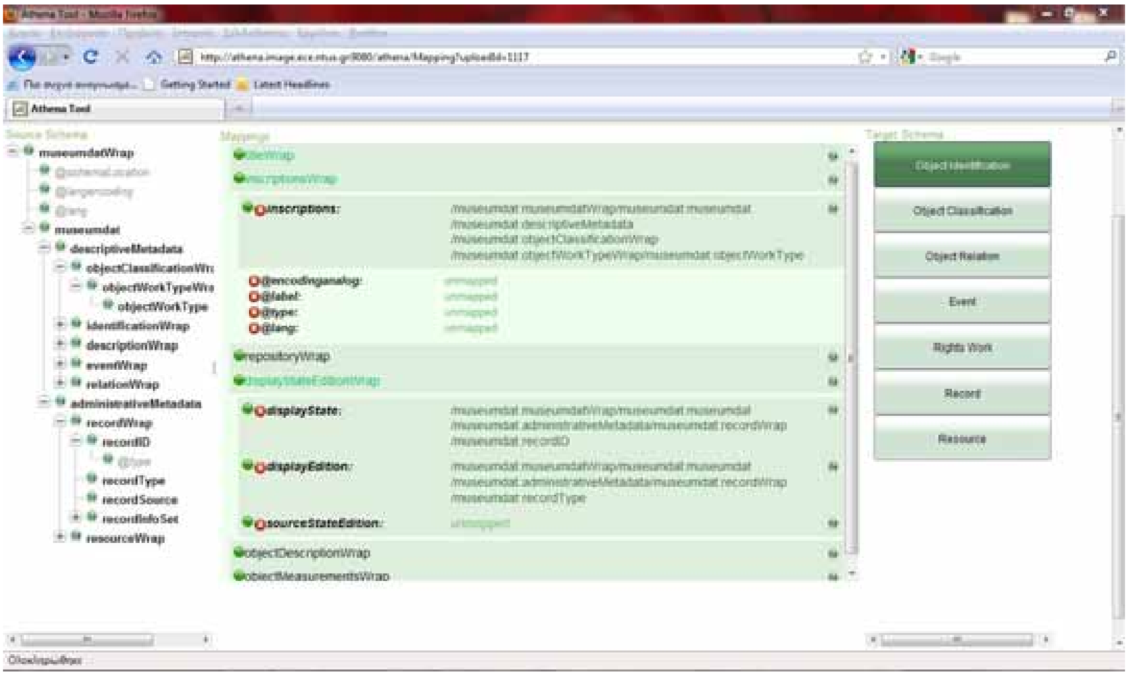


Figure 5.5 Message exchange between the mapping tool sub-modules.



inscriptions: /museumdat:museumdatWrap/museumdat:museumdat /museumdat:descriptiveMetadata /museumdat:objectClassificationWrap /museumdat:objectWorkTypeWrap/museumdat:objectWorkType

@encodinganalog: unmapped
@label: unmapped
@type: unmapped
@lang: unmapped

displayState: /museumdat:museumdatWrap/museumdat:museumdat /museumdat:administrativeMetadata/museumdat:recordWrap /museumdat:recordID

displayEdition: /museumdat:museumdatWrap/museumdat:museumdat /museumdat:administrativeMetadata/museumdat:recordWrap /museumdat:recordType

sourceStateEdition: unmapped

Figure 5.6 Screenshot of the mapping tool UI with partial mappings defined.

Software Interfaces

The mapping tool of provides and requests interfaces from various other modules of the ATHENA platform. More specifically there are the following interfaces:

- Interface from the XML Schema Parser sub-module to the User Interface sub-module.
- Interface from the XML Schema generator sub-module to the User Interface sub-module.
- Interface from the User Interface sub-module to the Item Summarization module.
- Interface from the Athena Data Layer to the User Interface sub-module.
- Interface from the Athena Data Layer to the XML Schema Parser sub-module.
- Interface from the Athena Data Layer to the XML Schema Generator sub-module.

Interface Name	Interface from the Athena Data Layer Module
Participant Module/Component	Athena mapping tool UI sub-module
Participant Module/Component	Athena Data Layer
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
Retrieves or saves the mappings defined by the user so far.	The UI sub-module communicates with the Athena Data Layer and stores or retrieves the mappings for a specific import. **Input:** A JSON representation of the mappings or the id of an import. **Output:** A status code or a JSON representation of the mappings of an import.

Interface Name	Interface from the Athena Data Layer Module
Participant Module/Component	XML Schema Parser sub-module
Participant Module/Component	Athena Data Layer
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
Retrieves the target XML Schema.	The XML Schema Parser requests the target XML schema in order to parse it and generate the appropriate data structures. **Input:** none. **Output:** The target XML Schema

Interface Name	Interface from the Athena Data Layer Module
Participant Module/Component	XML Schema Generator sub-module
Participant Module/Component	Athena Data Layer
Invoked/Invokable Methods	

Method	Description and Exchange Format/Type
Retrieves the data of a specified import in order to generate a tree representation of the Schema.	The XML Schema Generator requests the data of a specific import in order to generate a tree representation of the Schema. **Input:** The id of an import. **Output:** The data of that import.

Interface Name	Interface from the Athena mapping tool UI sub-module
Participant Module/Component	Item Summarization module
Participant Module/Component	Athena mapping tool UI sub-module
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
Invokes the mapping tool for a specific import.	The user is able through the item summarization module to invoke the Athena mapping tool for a specific import. **Input:** The id of an import. **Output:** none.

Interface Name	Interface from the Schema parser sub-module
Participant Module/Component	Athena mapping tool UI sub-module
Participant Module/Component	Athena mapping tool Schema parser sub-module
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type

Requests a JSON representation of the target Schema that is going to be rendered on the UI.	The UI sub-module requests from the Schema parser sub-module a JSON representation of the target XML Schema. **Input:** none. **Output:** A JSON representation of the target XML Schema.

Interface Name	Interface from the Schema generator sub-module
Participant Module/Component	Athena mapping tool UI sub-module
Participant Module/Component	Athena mapping tool Schema generator sub-module
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
Requests a tree representation using HTML of the generated XML Schema for a specific import.	The UI sub-module requests from the Schema generator sub-module a tree representation of the Schema generated for a specific import. **Input:** The id of a specific import. **Output:** A tree representation of the generated XML Schema in HTML markup for the requested import.

6 Statistics module

Implementation details

The ATHENA platform aims to be able to handle millions of metadata records that will be represented using the XML language. For many reasons the necessity of an easy way to inspect these metadata arises, mainly because it will be impossible for a user to inspect every single record without spending either too much time or making unavoidable mistakes. Also for many procedures that are part of the ATHENA workflow, e.g. quality assurance (QA) procedure, it is mandatory to design and implement a module that will be able to extract valuable information in the form of statistics from the imported metadata datasets. The basic design goal of the statistics tool is to implement a set of functions or methods that will be applied to the ingested metadata datasets. These methods will produce a set of metrics that will be valuable for the user in order to survey reliably and fast the metadata he wishes and also for the machine to be able to exploit the generated information in order to optimize various procedures that are part of the ATHENA workflow. In order to rationalize the decisions made for the design of the Statistics tool we have to describe the nature of the data we are attempting to analyze.

Statistics help providers understand and map their own data. Usually providers don't really know their input schema (they just export their database in xml format provided by their IT staff) and often are not sure of the specific values. Using statistics they can see what kind of annotations are in each field of their schema to decide how it has to be mapped. They can also be used to detect controlled lists. For example, from the statistic module you can see that a field called 'lang' is annotated in 200/200 items of an upload and takes 4 distinct values 'en', 'it', 'gr' and 'de'. You can detect mistakes as well, for example if you had 199 occurrences of the aforementioned values and 1 of the value 'tt'. Statistics are provided to each user for his own data only, there is currently no intention on publishing them or using them globally.

As it is mentioned above, the platform is going to store, handle and manipulate structured data and metadata that are described using the XML language. The XML Specification describes both syntax and a model. The syntax is based on angle brackets, while the model is basically a tree of nodes, which could be either elements, attributes or text containers. The XML model is described in detail in the XML Infoset specification provided by the W3C.

The ‘names’ of those nodes are not dictated by the XML model or syntax. XML is in fact a meta-language, a language to describe languages. In general there is no reason why nodes with the same name or content should be reused throughout the same XML tree, but in real life the number of different ‘names’ for those nodes is very small compared to the amount of data that the XML tree contains. In practice, the amount of different ‘names’ is a function of the different namespaces used by the XML tree and not a function of the size of the XML tree itself.

This is a basic characteristic of the XML language as it is used, upon which the generation of statistics makes sense while at the same time it provides valuable information on how to organize the generation of the statistics and which metrics should be used, as it is going to be presented in more detail later in this document. One example that clarifies the above statement is the following; the web contains billions of documents, and it is possible to convert all of them into XHTML and create a huge XML dataset several Terabytes big. Despite the size of this dataset, the structural complexity of that XML dataset would not be a function of its size, but a function of the schemas used, mainly XHTML.

Another interesting property that real life XML datasets exhibit is the fact that ‘semantic’ information (information that shows the user how to understand this) is not encoded in the node name but in the entire sequence of names that leads to that node. In brevity, in its XPath. An example of this is the use of the node “email” that has no meaning by itself, if not indicating that it contains an email address, but the path fragment “person/email” will indicate that this “email” node is contained inside the “person” node and for that reason, is associated to that. Real life analysis shows that the number of XPaths is a function of the complexity of the schema, therefore, again, not a function of the size of the dataset.

Apart from the above mentioned characteristics, in order to be able to start specifying the set of the methods and metrics needed to produce a complete and useful statistics for XML data, someone also has to make a few assumptions. The assumptions made in the case of the ATHENA platform are the following:

- The data is encoded in literals, either as textual nodes inside elements or as attributes values.
- The metadata is encoded in XPaths that identify those literals.
- Mixed content is considered as data.

Given those assumptions and the elements of the analysis of the basic characteristics of XML so far, we would like to be able to “inspect” the dataset and generate statistics that derive from the answers of the followings questions:

- What XPaths were used in this dataset?
- How many times were they used?
- What is the distribution of the values a specific element has?
- How many different schemas are used in a specific dataset?

The ATHENA platform follows a complete workflow for ingesting, managing, transforming and exposing metadata in various formats. For this reason the statistics that are useful are not only those related to the XML data. For this reason we need to define various levels of statistics that should be generated. These different levels are the following,

- Collection related statistics.
- XML Element statistics.
- Value Statistics.

Collection related statistics are those related to the ingestion process itself. For example, the number of distinct XML documents a specific user has ingested so far. XML Element statistics are those related to the questions stated earlier on this document. Finally, the value statistics refer mainly to the distribution of the various values of a specific XML element or attribute. Every set of statistics come in response to specific design needs of the workflow. The collection statistics for example, are needed for versioning and quality control. The XML element statistics are needed in the process of mapping and for inspecting the quality of the ingested metadata manually. Finally the value statistics are needed for ensuring the quality of the ingested metadata and provides an easy way to pinpoint errors on data, mainly on those controlled by specific vocabularies, but also needed in the process of normalization and enrichment.

One of the core design concepts of the statistics module is the readability and accessibility of the various statistics in a graphical appealing and intuitive way. From a software engineering perspective the architecture of the statistics module is based on principles that ensure reusability and scalability of the generated application.

The statistics module is separated in two distinct sub-modules. The first one is responsible for the generation of the statistics and the second one is responsible for the visual presentation of the statistics to the user. The user interface sub-module of the statistics module is written using JavaScript and is an application rendered and managed completely by the clients' browser. The generation of statistics sub-module runs on the server side and is integrated with the data layer. A proper interface is defined between the two sub-modules in order to maintain communication; this interface is based on AJAX calls made by the client to the server asynchronously. The rationale behind this architecture design is to decouple as much as possible the processing that happens to the server from the presentation part of the statistics module. In this way, the scalability and responsiveness of the whole system is ensured, especially when these principles are maintained in every step of the ATHENA workflow and the various applications that are defined in each one. Another advantage of this architecture is that it is possible to separate the implementation effort of the UI part and the core functionality that is needed. A consequence of this is that many changes can happen on the UI part in a consistent way, without affecting any other module or functionality of the system, so, also the extensibility of the statistics module is ensured. The statistics module architecture is presented in Figure 6.1.

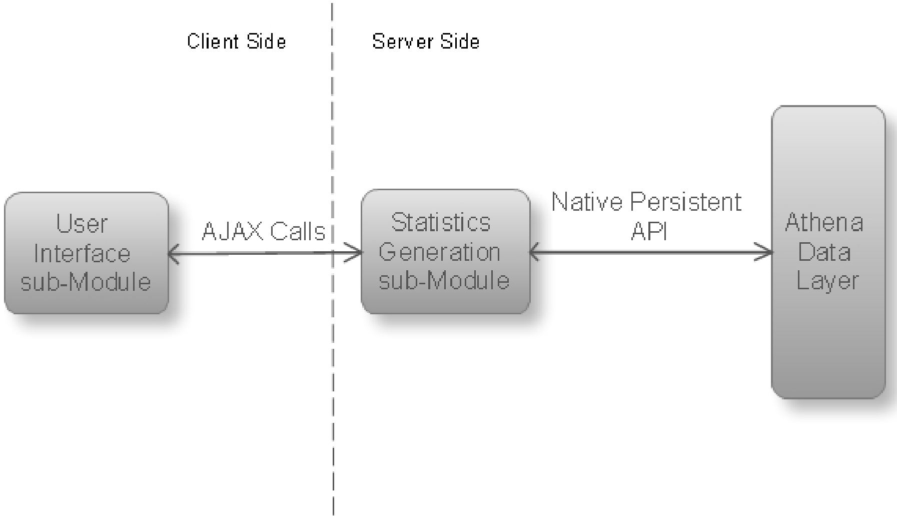


Figure 6.1 Architecture of the Statistics Module.

The statistics generation sub-module is mainly an object that implements a specific interface for requesting the required data from the database layer and calculate the appropriate statistics based on them. For this reason a native persistent API is used based on Hibernate in order to be able to access the ATHENA Data Layer and execute queries that will produce the requested statistics. The reason behind choosing a persistent technology based on Hibernate is to ensure platform neutrality from the various databases and schemas that could be used for storing data internally. By ensuring platform neutrality, also the extensibility of the statistics module is ensured.

Functionality and user interfaces

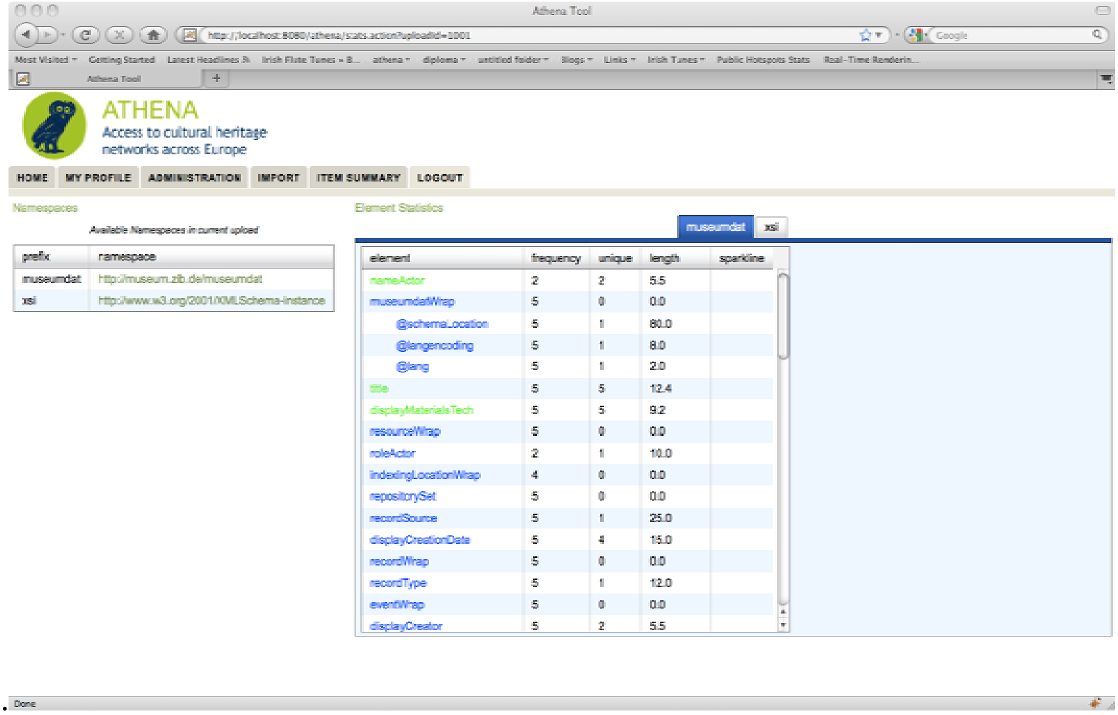
The User Interface sub-module is responsible to present to the user information from very large datasets in a consistent and visual appealing way. The user invokes the statistics module from the item summarization tab which is part of the web service interface and where the user is able to manage his imported metadata. Each import has a distinct button which when pressed transfers the user to a new screen (Figure 6.2) where the statistics user interface is rendered and the user is able to start inspecting the statistics of that particular import. A minimal set of information is transferred to the server when the user invokes the statistics tool; this information consists mainly of a unique id that identifies the specific import so the invoked module is able to load any data that are needed in order for the statistics to be calculated from the data layer.

When the user invokes the statistics module for a specific import, he is presented with the first set of statistics that refer mainly to the Schema and the XML Element information that can be calculated for that import. The screen is separated in two distinct areas. The first one from the right is used to present to the user the various XML schemas and prefixes for each of those that are present in this particular import. The second area of the first page consists of a view separated with tabs, for each XML Schema presented on the table that was described previously, also one tab is created that holds a table with information regarding the elements found in the specific import and that belong to that Schema. This table has 5 rows where statistics are presented, the names of the columns and their roles are the following:

- **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.
- **Frequency.** This is the frequency of a specific element compared to the number of distinct items that are present in the import.
- **Unique.** The numbers of distinct/unique values the element or attribute holds.
- **Length.** The average length of the values the element or attribute holds.
- **Sparkline.** A visual representation in the form of a graph that has the same size with the text font used in the table. This is used mainly to provide to the user a fast and visual way of understanding the distribution of the element or attribute in the import.

The user is able to distingue the elements from their corresponding attributes although both are presented in different rows of the table. This is achieved by positioning differently the elements from the attributes while grouping them together. The attributes of a specific element, always follow it in successive rows while an attribute always has the character @ in front of its name. Another visual aid of the statistics page is the usage of different colors that correspond to different characteristics that were found related to the data. These color codes together with their meaning are the following:

- Green. An element or an attribute has this color when the value of frequency equals the value of uniqueness. This characteristic might indicate a unique element or attribute that is or could be used as a possible identifier.
- Red. An element or an attribute has this color when every occurrence of it in the import had an empty string as a value. In this way the user is able to quickly identify any elements that are empty but they shouldn’t be.
- Grey. An element or attribute has this color when it is of type ComplexType. This indicates elements that should not hold any value but are wrappers of others.
- Blue. An element or an attribute has this color when none of the above criteria are present for them. Usually the majority of the elements that are supposed to hold a value should have this color code.



Available Namespaces in current upload

prefix	namespace
museumdat	http://museum.zib.de/museumdat
xsi	http://www.w3.org/2001/XMLSchema-instance

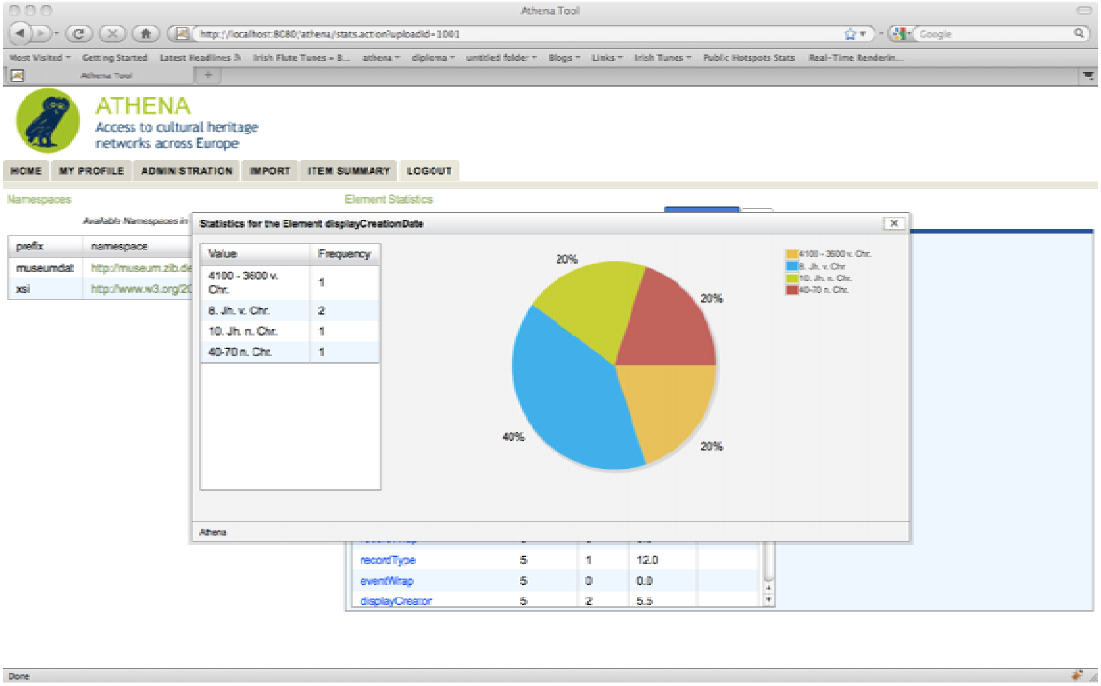
element	frequency	unique	length	sparkline
nameActor	2	2	5.5	
museumdatWrap	5	0	0.0	
@schemaLocation	5	1	80.0	
@langencoding	5	1	8.0	
@lang	5	1	2.0	
title	5	5	12.4	
displayMaterialsTech	5	5	9.2	
resourceWrap	5	0	0.0	
roleActor	2	1	10.0	
indexingLocationWrap	4	0	0.0	
repositorySet	5	0	0.0	
recordSource	5	1	25.0	
displayCreationDate	5	4	15.0	
recordWrap	5	0	0.0	
recordType	5	1	12.0	
eventWrap	5	0	0.0	
displayCreator	5	2	5.5	

Figure 6.2 Input schema statistics.

Another important set of statistics is related to the values of a specific element or attribute. These statistics are mainly used for the user to be able to inspect the content of a specific element or attribute and is only applicable for certain cases where the length of the values do not exceed a certain threshold that is defined. This threshold exists for two reasons. The first reason is that while the median length of the value increases, so do the chances that the set of value lacks any coherency, so by presenting all the values would result in a huge view of data while not offering any valuable information to user. The second reason is to reduce the computational overhead on the server. Also for optimizing the performance both on the client and the server an interface based on AJAX calls is implemented that retrieves the data related to a specific element or attribute when the user double clicks on one of them as they are presented on the first page of the statistics user interface sub-module. In this way it is possible to reduce the memory footprint on the client side while at the same time reduce the computational overhead of calculating statistics for every possible element or attribute, on the server side. When the user wishes to view statistics related to the values of a specific element or attribute a modal window pops up separated in two distinct parts (Figure 6.3). The right part contains an HTML table enhanced with functionality using JavaScript using the YUI library that has the following two columns.

- **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.

The left part of the modal window presents the data of the table in a visual way using a pie chart. In the future the user will be able to select dynamically the type of diagram he wishes to view, e.g. bar diagram instead of a pie chart. When the user decides that he finished inspecting the value distribution of a specific element or attribute he is able to close the modal window and return to the first view of the statistics User Interface sub-module. All the data are retrieved from the server using AJAX calls and rendered in the client side for performance reasons.



prefix	namespace
museumdat	http://museum.zib.de
xsi	http://www.w3.org/20

Value	Frequency
4100 - 3600 v. Chr.	1
8. Jh. v. Chr.	2
10. Jh. n. Chr.	1
40-70 n. Chr.	1

recordType	5	1	12.0
eventWrap	5	0	0.0
displayCreator	5	2	5.5

Figure 6.3 Value distribution statistics for a specific element or attribute.

Software Interfaces

The statistics module of the Athena platform provides and requests interfaces from various other modules of the platform. More specifically there are the following interfaces:

- Interface from the Athena Data Layer Module.
- Interface to the Athena Item Summarization Module.
- Interface to the Athena Mapping Tool Module.

Interface Name	Interface from the Athena Data Layer Module
Participant Module/Component	Athena Statistics Module
Participant Module/Component	Athena Data Layer
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
Retrieves a set of pre-calculated statistics as requested by the Athena Statistics Module	The Athena data layer module executes a set of queries and returns a set of statistics as requested by the Athena Statistics module. **Input:** the name of an element, attribute, namespace or import. **Output:** Statistics generated by the execution of the appropriate queries on the Athena database.

Interface Name	Interface to the Athena Item Summarization Module
Participant Module/Component	Athena Item Summarization Module
Participant Module/Component	Athena Statistics Module
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
The user is able from the Item Summarization Module to invoke the Athena Statistics Module in order to be presented with the statistics for a specific import.	The Athena Statistics Module is invoked by the user from the Item Summarization Module for a specific import. **Input:** The ID of a specific import. **Output:** A set statistics for that import

Interface Name	Interface to the Athena Mapping Tool Module
Participant Module/Component	Athena Mapping Tool Module
Participant Module/Component	Athena Statistics Module
Invoked/Invokable Methods	
Method	Description and Exchange Format/Type
The Athena Mapping tools is able to retrieve a minimal set of statistics useful for the procedure of mapping.	The Athena Mapping tool requests a minimal set of statistics for a specific element from the Athena Statistics module in order to assist the user in the process of mapping his schema to the LIDO Schema. **Input:** The name of a specific element or attribute. **Output:** A minimal set of statistics for this particular element or attribute.

7 Relevant 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 3.1.

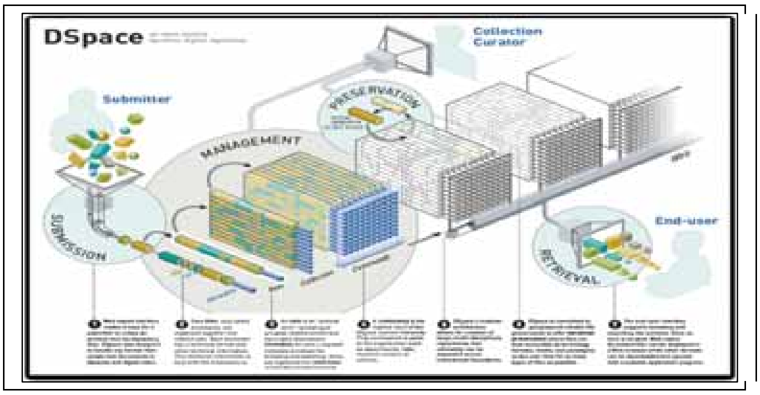


Figure 3.1 the over architecture of the DSpace platform.

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 reimplementation 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.

- DSpace should be primarily open source software for building digital repositories.
- DSpace should be usable based purely on free and open source software.
- DSpace should have a decoupled, stable and application neutral core.

- 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.
- DSpace should employ and support existing, open standards where possible and practical.
- DSpace releases should be minimal disruptive.
- DSpace should support an exit strategy for content.
- 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 us policy enforcement, and performance enhancement to support very large repositories. The system’s interface comprises three web based services:

- A management API that defines an interface for administering the repository, including operations necessary for clients to create and maintain digital objects;
- An access API that facilitates the discovery and dissemination of objects in the repository; and
- 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:

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

REPOX – A Metadata Space Manager

Repox 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 Repox 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. Repox is developed in Java so it can be deployed in any operating system that has an available Java Virtual Machine.
- Easy deployment. Repox is available with an easy installer, which includes all the required software and libraries.
- Support for several metadata formats. Repox 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.

Repox 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 Repox is limited to support only the exposure of metadata transformed in the format defined and supported by the TEL project.

8 Ongoing Development

Heading towards the actual ingestion and publishing of content through the ATHENA project, WP7 compiles a list of feature requests and feedback from the tool's usage that will finalize the prototype of the semantic interoperability platform. There are several issues and additional functionalities that will be considered, including, but not limited by, the following:

- Incorporate a fully functional OAI-PMH scheduling and harvesting interface
- Input data source management and versioning
- Enhance visual aid and user guiding functionalities based on standardised schemata and statistical information
- Support established metadata standards for the implementation of semi-automatic transformations
- Implement a visual editor for additional xslt-supported functions
- Register and look up controlled vocabularies associated with providers' schemata
- Enhance statistical analysis
- Support optional element duplication based on user selection

WP7 is closely following all developments in the Europeana family of projects and the tool is prepared to support additional schemas for exposing metadata as well as the formation of digital objects according to the developments concerning the Europeana Data Model (EDM).

9 Conclusions

Present report documents the first version of the semantic interoperability platform that is designed and implemented within WP7 of the ATHENA project. The web service is used to ingest metadata from a diverse group of cultural heritage content providers, homogenise and align them with an established metadata schema standard that guarantees semantic interoperability and, publish them in the Europeana Semantic Elements (ESE) schema for harvesting and presentation in the Europeana portal. It is based on a platform developed by the leader of the WP, which is customised and updated with all current state of the art technologies and the most recent developments in the Europeana family of projects. Functionality includes a user and organisation management scheme that supports appropriate user roles and access rights for simple organisations up to the formation of complex thematic or national aggregators, import of arbitrary metadata schemata used by providers and serialized in xml, a statistical module for input data sources, a visual mapping module that functions as an xslt editor from a data source to a reference metadata schema that enables semantic interoperability, transformation of imported data sources, publishing and exposing aggregated metadata in standard metadata schemata, focusing on Europeana enabled content.

The tool was presented and tested by a wide variety of ATHENA users that attended a respective WP7 workshop and is in the process of incorporating additional functionalities based on user feedback and recent developments in the field of digital cultural heritage and the web of data. Ingestion process will initiate in regional workshops that will be organised within ATHENA in order to register and train providers. Ingestion will be supported continuously until the end of the project. Following the first ingestion phase that is marked by the Europeana Rhine release, the tool will be re-evaluated and finalized for the final version of the prototype.