Digital Repository for Life-long Competence Development

A. Grigorov\textsuperscript{1,2}, A. Georgiev\textsuperscript{1}, M. Petrov\textsuperscript{1}, K. Stefanov\textsuperscript{1}

\textsuperscript{1} Faculty of Mathematics and Informatics, Sofia University
\textsuperscript{2} Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
alexander.grigorov@fmi.uni-sofia.bg, atanas@fmi.uni-sofia.bg, milenp@fmi.uni-sofia.bg, krassen@fmi.uni-sofia.bg

Abstract: This paper discusses the building of a digital knowledge repository for life-long competence development. The repository is an essential part of the LearnWeb2.0 system, which is aimed at stimulating knowledge sharing, knowledge management and the transformation of information into knowledge. The paper describes system’s architecture, the choice of a digital repository, the modelling of digital objects and the metadata for resources.

Keywords: Digital Repositories, Web 2.0 services, Knowledge Resource Sharing and Management (KRSM), Knowledge Object Models, Knowledge Resource (KR)

1 Introduction

Within the TENCompetence project we have built an open source system LearnWeb2.0 for stimulating knowledge sharing, knowledge management and the transformation of information into knowledge into communities of practices (Marenzi et al., 2008). Essential parts of the system are the knowledge repository and the Knowledge Resource Sharing and Management (KRSM) web services, which allow access and management of the repository.

In this article we discuss the following research questions and tasks related to the building of the knowledge repository:

- Designing a multi tier architecture for the LearnWeb2.0 system.
- Selecting a digital repository that best meets the requirements for life-long competence development.
- Designing and implementing appropriate digital object models.
- Designing and implementing web services for knowledge resource sharing and management that serve the needs of the TENCompetence project.
- Using metadata standards for describing resources.

Before discussing them we present a short overview of the most popular currently available digital repositories.

2 Digital Repositories and Related Projects

One of the most popular digital repositories is DSpace (Smith et al., 2003). It was originally designed by developers at the MIT Libraries and HP Labs and currently is used by over 250 institutions. DSpace™ is a free, open source software platform for building repositories of digital assets, with a focus on simple access to these assets, as well as their long-term preservation (Smith et al., 2003). It was originally designed with a particular service model in mind: that of institutional repositories of research material, and particularly research articles, which were produced by academic research
institutions. A drawback of DSpace is that it uses a fixed web interface and cannot be easily integrated in other systems.

Another example is the Knowledge Pool System ARIADNE (Duval et al., 2001). It was built by an European educational digital library project initiated in 1996 by the “Telematics for education and training” program of the European Commission. It consists of a distributed digital library of educational resources aimed at delivering reusable components to teachers and learners from different cultures and with different languages. The most innovative aspect of ARIADNE was its metadata. The new aspect that this project proposed was the semi-automatic generation of this metadata. Since the typical end user of this system was expected to be a teacher, this process should be simple and easy.

Another popular repository system is Fedora (acronym for Flexible Extensible Digital Object Repository Architecture) - an open source, digital object repository system (Payette and Lagoze, 1998). Using a standards-based, service-oriented architecture, the Fedora platform provides an extensible framework of service components to support features such as OIA-PMH, search engine integration, messaging, workflow, format conversion, bulk ingest, and others. In addition, features such as authentication, fine-grained access control, content versioning, replication, integrity checking, dynamic views of digital objects, and more are incorporated into the Fedora repository service (Grizzle et al., 2004, Lagoze et al., 2006, Staples et al., 2003).

Fedora has been adopted by hundreds of institutions for an array of innovative applications including open-access publishing, scholarly communication, e-science, digital libraries, archives, education, and more.

The RepoMMan Project (Green et al., 2007) is developing a tool which will allow users to interact with a Fedora digital repository as part of their natural workflow. The University of Hull takes a broad view of repository functionality, seeing it as offering storage, access, management and preservation of a wide range of objects from conception to completion and possible publication. The effectiveness of a repository is linked to the quality of its metadata.

The University of Virginia Library is attempting to solve four problems with their Fedora implementation (Fedora Commons UV, 2009):

- management of complex objects organised in complex hierarchical structures
- management of objects having highly disparate data types and preservation requirements
- organisation of data objects in virtual collections by identifying and presenting in the repository the relationships between them
- collection of born-digital faculty projects that incorporate new and reused materials into new scholarly contexts.

Fedora was chosen by the University of Virginia staff because it was architected to facilitate handling of complex objects (Fedora Commons UV, 2009).

3 LearnWeb2.0 Functionality and Architecture

LearnWeb2.0 system is aimed at stimulating knowledge sharing, knowledge management and the transformation of information into knowledge. The main functionalities of the final version of the system include:

- Search for available resources from a number of existing Web 2.0 tools (such as YouTube, Flickr, etc.);
- Search for resources in the TenCompetence Fedora repositories (search by keywords, by categories, by tags);
- Upload and store resources and metadata in Fedora repository;
- Metadata editor;
- Commenting, tagging and rating of resources;
• Category management and organization of resources in categories;
• User management and registration;
• User authentication and authorization;
• My home page for users;
• User stimulation by ranking;
• Connectivity with other TenCompetence tools (Liferay, TenC server, WebPDP, TenTube, etc.);
• Browser toolbars;
• Etc.

A simplified scheme of the LearnWeb2.0 architecture is shown in Figure 1. The main components of the systems are:
• The LearnWeb2.0 Web Tool (written in PHP using the CakePHP framework) - for interactive management of knowledge resources;
• KRSM Web Services (implemented in Java) - for automatic management of knowledge resources;
• InterWeb services (written in PHP) – for searching and retrieving resources from Web 2.0 tools;
• Fedora repository (the last version of LearnWeb supports multiple Fedora repositories);

![LearnWeb2.0 Architecture Diagram](image)

**Figure 1. The LearnWeb2.0 Architecture.**

Other TenCompetence tools and servers (for example the PCM Server, ReCourse, etc.) access the knowledge repository through the KRSM Web Services.

We have chosen Fedora as a basic repository platform for the following reasons:
• Fedora supports flexible and extensible digital objects, which are containers for metadata, one or more representations of the content and relationships to other information resources.
• Fedora's digital objects provide building blocks to support uniform management and access to heterogeneous content including books, images, articles, datasets, multi-media, and more.
• Fedora is implemented as a set of web services that provide full programmatic management of digital objects as well and search and access to multiple representations of objects.
• Fedora is particularly well suited to exist in a broader web service framework and act as the
foundation layer for a variety of multi-tiered systems, service-oriented architectures, and end-user applications.

The architectural view of the Fedora digital object model is shown on Figure 2 (Grizzle et al., 2004).

![Fedora Digital Object Model](image)

*Source: Grizzle et al. (2004)*

**Figure 2.** Fedora Digital Object Model

Access to the digital object is provided by disseminators, which can simply deliver a desired portion of the digital object or can deliver a customized view. Fedora’s digital objects are self-describing and have self-delivering-key features that enable preservation.

## 4 Digital Object Models

We have identified and defined the following types of digital objects:

- **User** – an object representing a person who uses the system.
- **Category** – an object containing other categories and/or resources.
- **Resource** – a resource stored on the server. Each resource has metadata in Dublin Core format (Dublin Core, 2009) that describes the resource. The content of the resource can be stored on the server or anywhere on the Web (in this case the URL of the resource is stored on the server). Resources have tags, comments, popularity and rating.
- **Tag and Tagging** – objects used for tagging resources. The Tagging object connects a user, a resource and a tag.
- **Comment** – an object used for commenting resources. The comments can be rated by users.

The designed Fedora Digital Object Models and the relations between them are shown in Figure 3. Each object is represented as a digital object in Fedora with corresponding datastreams. The relations between the objects are represented and implemented by defining appropriate Fedora relationships. A number of methods are also defined for extracting information about the objects by creating several Behavior Definition Objects and Behavior Mechanism Objects.
Figure 3. Digital object models with relations.

Figure 4 shows an example of the Digital data model for resources. A resource has a PID (Persistent ID) and the following datastreams:

- **DC** - Dublin Core metadata.
- **REL-EXT** – a datastream containing the external relations of the resource with other digital objects expressed in RDF.
- **Link** – a datastream containing the URL of a Web resource.
- **Rating** – a datastream containing the current rating of the resource.
- **Score** – a datastream containing the popularity of the resource.
- **Content** – a datastream for storing the content of the resource.

The relations defined between the objects include:

- **owner** – a relation between a resource and an user;
- **isMemberOf** – a relation between a resource and category, stating that the resource belongs to the category;
- **isSubsetOf** – a relation between categories used to organize the categories in a hierarchy;
- **isCommentOf** – a relation between a comment and a resource;
- **isRatedBy** – a relation between a resource and an user, stating that the user has rated the resource.

Since the relationships in Fedora are binary, for expressing the 3-nary relationship user-resource-tag (an user has rated a resource with a tag), we have introduced a Tagging object that connects the user, the resource and the tag.

5 KRSM Web Services

The LearnWeb2.0 system will be built on the base of the protocols and standards from the SOA (2009). According to this architecture, the main system components are web services, which communicate with each other using well known standard protocols such as WSDL (SOA, 2009). All the software developed in the project uses relevant web and e-learning standards and specifications such as SOAP, UDDI, XML, XSLT, RSS, RDF, IMS LD, IMS CP, IMS QTI, IEEE LOM, etc.

Our approach is also known as loose coupling of components and applications. Loose coupling can be differentiated regarding the level of integration. At the very basic level, a loose coupling is making two applications available on the user’s desktop. The fact that these applications share common data may provide sufficient coupling between them. At the other end of the spectrum loose coupling will be achieved by using a SOA. LearnWeb2.0 provides both types of loose coupling.

The advantages of a SOA are:
- changes to the implementation of the service should not affect the service consumer
- the service consumer is free to choose an alternative service without modifying their application, except the address of the new service
- web services don’t require a service consumer and provider to use the same technologies, except that both need to use the same web service protocols.

In order to implement the loose coupling type of SOA integration, LearnWeb2.0 will use a web portal providing the User Interface to all well-defined functionalities and tools. This web portal will provide the required integration through the user interface offered.

The web services exposed by Fedora include (Payette and Lagoze, 1998):
- Fedora access service (SOAP, REST)
- Fedora management service (SOAP, REST)
- Basic search: repository search (REST)
- Basic OAI: simple OAI-PMH provider (REST)
- OAI provider service: a configurable OAI-PMH provider (REST)
- RISearch: resource index search (REST)

The WSDL definitions for these services are available at the Fedora site.

The web services used in LearnWeb 2.0 are called KRSM services. KRSM web services are developed in Java and the APIs for the services are modelled using the REpresentational State Transfer (REST) approach (SOA, 2009). The implemented web services (over 60) are divided into two groups:
- Access-API-Lite
- Management-API-Lite.

The Access-API-Lite services are used for retrieving information and metadata about: Resources, Categories, Users, Tags, Ratings, and Comments.

These services also implement integrated search for resources in the Fedora repository and in Web 2.0 tools using the corresponding adapters (drivers).
The Management-API-Lite services are used for creation and modification of resources, users, categories, tags, etc.

The KRSM web services use XML for exchanging information. We have defined XML schema for each type of objects stored in the repository. Figure 5 shows an example of the XML used for a resource.

The implemented Web services are intensively used by the LearnWeb2.0 web application. They can also be used for knowledge resource sharing and management by applications developed within the TENCompetence project.

```
<resource>
  <canonical>
    <link
    <dc>
      <title>What Is Web 2.0</title>
      <description>A paper about Web 2.0</description>
      <identifier>resource:1</identifier>
      <creator>Tim O'Reilly</creator>
      <format>text/html</format>
      <subject>Web 2.0</subject>
      <type>HTML document</type>
      <date>06/10/2008</date>
      <language>English</language>
    </dc>
    <category>
      <link href="category:22" type="category">category:22</link>
      <title>KRSM Category 2</title>
    </category>
    <owner>
      <link href="user:1022" type="user">user:1022</link>
      <title>Ruud</title>
    </owner>
    <tag-list>
      <tag>
        <link href="tag:2" type="tag">tag:2</link>
        <title>paper</title>
      </tag>
    </tag-list>
    <rating>
      <value>47</value>
      <numberOfVotes>15</numberOfVotes>
    </rating>
    <score>5</score>
    <comments>
      <count>4.0</count>
    </comments>
  </canonical>
</resource>
```

Figure 5. An example of the XML for a resource.

6 Resource Metadata

We have chosen to use the Dublin Core (DC) metadata standard to express the metadata for resources for the following reasons:

- Most of the knowledge resources used in the project can be fully described using DC.
- The Fedora repository has full support for DC: automatically creates indexes on DC fields and supports search within DC fields.
- Fedora allows easy extending of the metadata with custom fields.
The DC standard defines a simple yet effective element set for describing a wide range of networked resources. The Dublin Core standard includes two levels: Simple and Qualified. LearnWeb2.0 uses the Simple Dublin Core, which comprises fifteen elements.

LearnWeb2.0 uses resource metadata for searching and discovering resources and for proper view and manipulation of resources. We have designed and implemented a Metadata Editor corresponding to the specific data model of LearnWeb2.0. The editor is a web based application written in PHP using the CakePHP framework. It uses the KRSM web services and is integrated in the LearnWeb2.0 web tool.

In LearnWeb2.0 the owner of a resource stored in the Fedora repository is responsible for supplying the metadata for the resource. Only the owner can use the Metadata Editor to fill in the values of Dublin Core elements.

When the user adds a Web resource or uploads a resource to the repository she/he also has to provide metadata for the resource using the Metadata Editor.

7 Conclusion

In this paper we have discussed our approach for building a knowledge repository for storing, searching, accessing and retrieving knowledge resources for life-long learning within the TENCompetence project. We have selected Fedora as a basic platform for the repository and have designed and implemented suitable digital data models. We have also implemented web services for knowledge resource sharing and management.

LearnWeb2.0, the repository and the services have been iteratively developed and thoroughly tested to prove their functionality. The tool has been included in several TenCompetence pilots and also used by business demonstrators. Identified issues have been solved and improvements based on user feedback have been implemented in the final version of LearnWeb2.0.

Acknowledgment

This work has been funded by the European Commission through the TENCompetence project (IST-2004-02787).

References


Fedora Commons web site (2009), University of Virginia Library Use case, Available online at http://www.fedora-commons.org/usecases/libraries.php?pid=UVA


