Building a Knowledge Repository for Life-long Competence Development

A. Grigorov¹,², A. Georgiev¹, M. Petrov¹, S. Varbanov², K. Stefanov¹
1) Faculty of Mathematics and Informatics, Sofia University, Bulgaria
2) Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria
{alexander.grigorov, atanas, milenp, varbanov, krassen}@fmi.uni-sofia.bg

Abstract
This paper focuses on building a knowledge repository for life-long competence development. It is an essential part of LearnWeb2.0 system designed for stimulating knowledge sharing, knowledge management and the conversion of information into knowledge. The paper discusses the system architecture, the choice of a digital repository, the modelling of digital objects and the metadata for resources.

1. Introduction
Within the TENCompetence project we are building an open source system LearnWeb2.0 [5] for stimulating knowledge sharing, knowledge management and the conversion of information into knowledge into communities of practices.

Essential parts of the system are the knowledge repository and the KRSM (Knowledge Resource Sharing and Management) web services which allow access and management of the repository.

In this paper we discuss the following research questions/tasks regarding the building of the knowledge repository:
• design a multitier architecture for LearnWeb2.0 system;
• select a digital repository that best meets the requirements for life-long competence development;
• design and implement appropriate digital object models;
• design and implement web services for knowledge resource sharing and management that serve the needs of the TENCompetence project;
• use of metadata standards for describing resources.

2. Digital Repositories and Related Projects
One of the most popular digital repositories is DSpace [7]. 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 [7]. It was originally designed with a particular service model in mind: that of institutional repositories of research material, and particularly research articles, which are 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 [2]. It was an European educational digital library project initiated in 1996 by the European Commission’s telematics for education and training program. It consist on a distributed digital library of education resources delivers reusable components to teachers and learners form 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-automatically generation of this metadata. Since the typical
end user of this system was thought to be a teacher, this process should be simple and easy.

Another popular repository system is Fedora[1] - an open source, digital object repository system [6]. 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 [3, 4, 8].

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 [1] 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 function, 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: management of complex objects that are organized in potentially multiple hierarchical structures; management of highly disparate data types and their preservation requirements; building virtual collections by recording and identifying relationships between objects in the repository; and the collection of born-digital faculty projects that incorporate new and reused materials into new scholarly contexts. Fedora was chosen because it was architected to facilitate handling of complex objects [9].

### 3. LearnWeb2.0 Architecture

A simplified scheme of the LearnWeb2.0 architecture is shown on figure 1. The main components of the systems are:

- LearnWeb2.0 Web Tool (written in PHP using CakePHP framework) - for interactively manage knowledge resources;
- KRSM Web Services (in Java) - for automatically manage knowledge resources;
- Fedora repository;

Figure 1. LearnWeb2.0 Architecture

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

We have chosen Fedora as a basic repository platform because:

- 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, multimedia, 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 Fedora digital object model is shown on Figure 2 [3].

---

1 *Fedora* is an acronym for Flexible Extensible Digital Object Repository Architecture [6]
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 self-delivering—key features that enable preservation.

4. Definition of Digital Object Models

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

- **User** – a person who uses the system;
- **Category** – contains other categories and/or resource;
- **Resource** – a resource stored on the server. Each resource has metadata in Dublin Core format [10] 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** – used for tagging resource;
- **Comment** – for commenting resources. The comments can be rated by users.

The designed Fedora Digital Object Models and the relations between them are shown on 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 4 shows an example of the Digital data model for resources. The resource has a PID (Persistent ID) and the following datastreams:

- **DC** - Dublin Core metadata;
Since the relations in Fedora are binary, for expressing the 3-nary relation that 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 KRSM Web Services are developed in Java and the APIs for the services are modelled using the REpresentational State Transfer (REST) approach. The implemented web services (currently 44) are divided into two groups:

- Access-API-Lite (27 services)
- Management-API-Lite (17 services).

The Access-API-Lite services are used for retrieving information and metadata about:

- Resources;
- Categories;
- Users;
- Tags;
- Ratings;
- Comments;
- etc.

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 object stored in the repository. Figure 5 shows an example of the XML used for a resource.

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

6. Resource Metadata

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

- most of the knowledge resources used in the project can be fully described using DC;
- Fedora repository has full support of DC, automatically creates indexes on DC fields and supports search within DC 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/discovering resources and for proper view/manipulation of the 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 the 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.

Dublin Core metadata is sufficient to describe most of the resources used within TENCompetence project. However, some types of resources, such as Learning Objects, Learning Designs, etc., need additional metadata usually described in the Learning Object Metadata standard (LOM). The next version of LearnWeb2.0 will be extended to support LOM for such types of resources.

7. Conclusion and Future Work

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 designed and implemented suitable digital data models. Also we have implemented web services for knowledge resource sharing and management. The repository and the services have been thoroughly tested and have proved their functionality. Some issues have been identified that are being solved for the next version.

We are planning to extend the described above digital object models to support user groups, LOM metadata, advanced search and to define improved access to the resources. The following types of resources will be supported:

- **Public** – the resources can be accessed by all users;
- **Private** – the resource can be accessed only by the owner;
- **Shared** – the resources is shared to a group of users.

**Acknowledgment**

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

**References**


