D1.1 Open ICOPER Content Space Implementation of 1st Generation of Open ICOPER Content Space including Integration Mini Case Studies

ICOPER

ECP 2007 EDU 417007

ICOPER

D1.1 Open ICOPER Content Space Implementation of 1st Generation of Open ICOPER Content Space

including Integration Mini Case Studies

Deliverable number	*D-1.1*
Dissemination level	*Public*
Delivery date	*28 February 2009*
Status	*Final*
Author(s)	*Michael Totschnig (Editor), WUW* *Joris Klerkx, KULeuven* *Tomaž Klobučar, JSI* *Effie Law, ULE* *Bernd Simon, WUW* *Stefaan Ternier, OUNL*

e*Content*plus

This project is funded under the *e*Content*plus* 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 Introduction

In the context of the ICOPER project, the Open ICOPER content space (OICS) has been defined as the umbrella combining a portfolio of interoperable repositories, content and tools, as a test bed for the specifications and standards that become part of the ICOPER reference model. The use cases that the OICS will support emerge from the collaboration of all work packages. After fruitful discussions between all partners of work package 1, it was decided to conceive the first generation of the OICS as a meta-repository providing search functionality. Technologically, it would be based on metadata harvesting and provide a simple search interface for querying this metadata. This should build the foundation for the implementation of more sophisticated use cases that are currently collected by work package leaders, and are analyzed under the supervision of WP7. These use cases serve as concrete inputs for the future development of OICS. Detailed descriptions about this task are presented in Section 7.

This deliverable has been realized in the form of a prototype that has been made available to the consortium members under http://test1.km.co.at/OpenICS. This document describes the technical architecture of this prototype and enumerates the content that has been successfully imported.

2 Technological architecture

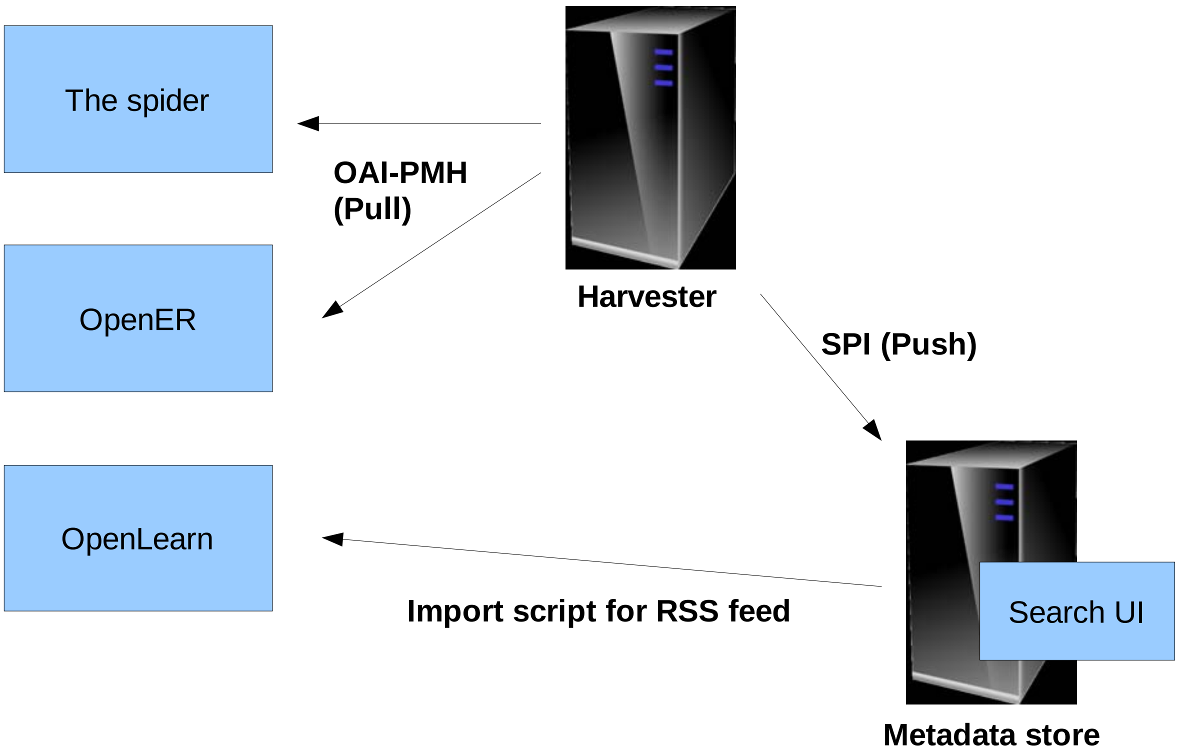

Figure 1: Architecture of Open ICOPER content space prototype

Three types of components are connected in the architecture chosen for this prototype (see Figure 1):

1. Three repositories have made their metadata available. Two of them (The spider, OpenER) provide an OAI-PMH interface, OpenLearn exposes metadata through an RSS feed that can be used to retrieve separate metadata files for each learning resource.
2. The OAI-PMH harvesting framework developed by KU Leuven, pulls metadata from the OAI-PMH targets, and publishes it via an SPI service into the metadata store.
3. The metadata store is based on the OpenACS framework. Besides accepting input through SPI, an import script reads the OpenLearn RSS feed, retrieves the metadata files, converts metadata to LOM and ingests it into the database. A simple user interface (UI) is provided for searching the metadata store.

The prototype is enabled by the following standards:

1. IEEE LOM (Learning Object Metadata) is used for representing metadata about learning objects (LOM, 2009).
2. OAI-PMH (Open archives initiative – Protocol for metadata harvesting) permits selective, incremental harvesting of two repositories (OAI, 2009).
3. RSS (Rich site summary) exposes identifiers from one repository, which can be used to retrieve metadata in a custom format.
4. SPI (Simple publishing interface) connects the harvesting framework to the metadata store (SPI, 2009).

2.1 OpenACS and the new-lors package

For the first prototype of the Open ICOPER content space, it was decided to use the application development platform OpenACS that was already successfully used by the Wirtschaftsuniversität Wien and its partner KnowledgeMarkets as the basis of two repositories of learning content (Bildungspool (Bildungspool, 2009), the federation of learning content for Austrian schools, and Educanext (Educanext, 2009), the brokerage platform for educational content).

OpenACS “is a toolkit for building scalable, community-oriented web applications.” (OpenACS, 2009). It relies on AOLserver, an open-source, multithreaded, scalable, Tcl-enabled, web/application server and the Relational Database Management System (RDBMS) PostgreSQL. On top of OpenACS, KnowledgeMarkets has developed a learning object repository service, capable of storing learning objects and metadata in configurable schemas, called *new-lors*. The new-lors package makes use of many of the services provided by the OpenACS platform (see Figure 2), most prominently the content repository and XoTcl, an object oriented extension to the Tcl programming language (XoTcl 2009).

For the next generation of the OICS, interfaces will be defined between this platform and the systems provided by the other technology providers of the ICOPER consortium.

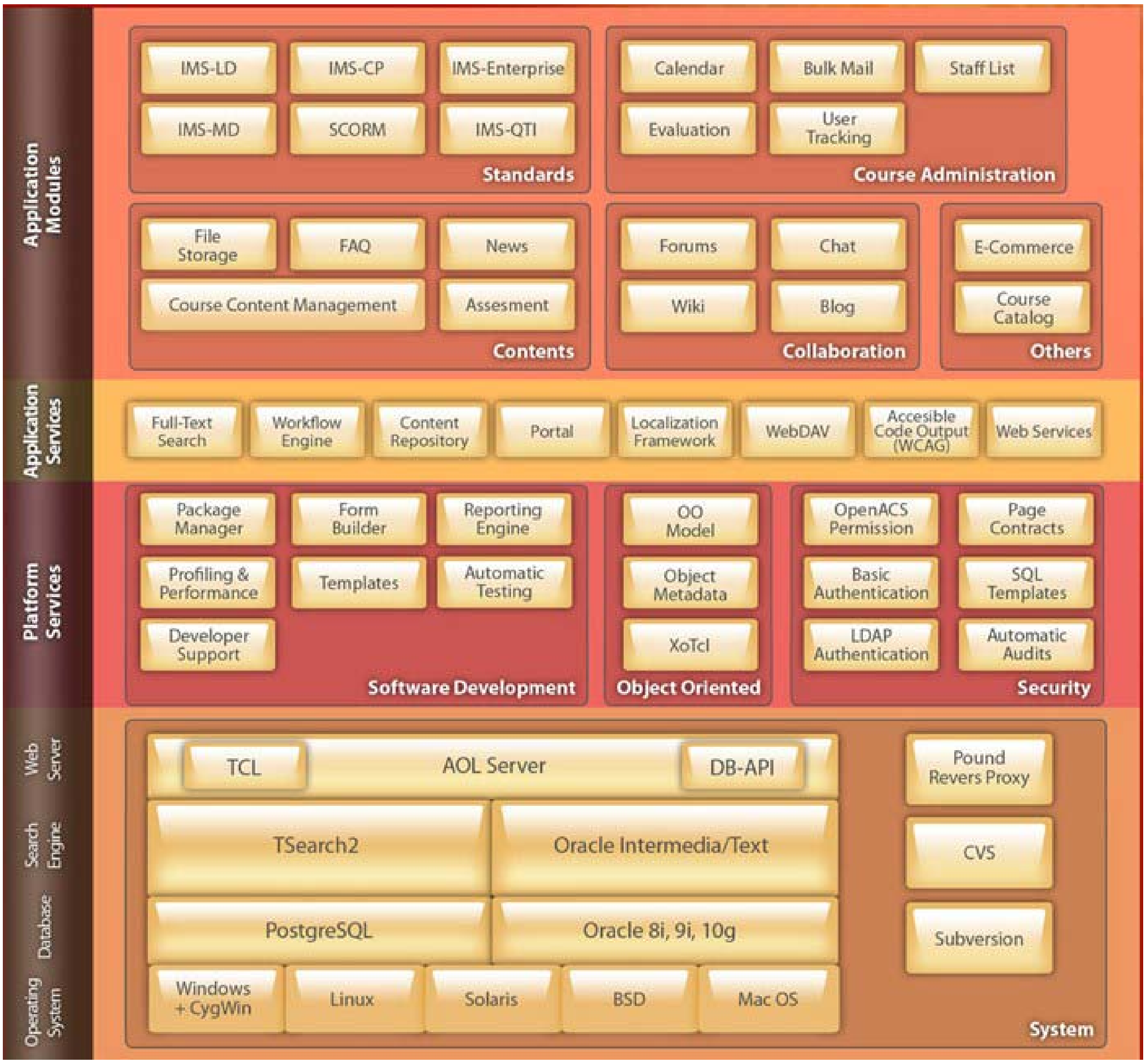

Figure 2: OpenACS Architecture (DotLRN 2009)

2.2 OAI-PMH harvester

K.U. Leuven has developed an OAI-PMH harvester, which is used in the current prototype system for harvesting metadata from some of the participating repositories. It is capable of forwarding the retrieved metadata via the SPI protocol that has also been implemented into the new-lors repository.

3 Features

The user interface of this prototype is deliberately kept simple, in order to remain open in many different directions, once work package leaders have agreed upon the use cases that the next generation of the OICS will support.

3.1 LOM storage

Since LOM is the established standard for encoding metadata about learning objects, it was essential to implement it as the basis for the first prototype of the OICS. The *new-lors* package did not yet provide support for the LOM metadata schema, but not more than the simple definition of a schema file was necessary in order to store metadata in LOM schema.

3.2 User interface

Whereas the backend implementation for LOM metadata is nearly complete, only little effort has been invested in the user interface, since the detailed use cases that the next generation of the OICS is supposed to support, are still discussed by all work packages of the project. Thus the current prototype deliberately has a very simple access page (see Figure 3). Its elements are briefly described below.

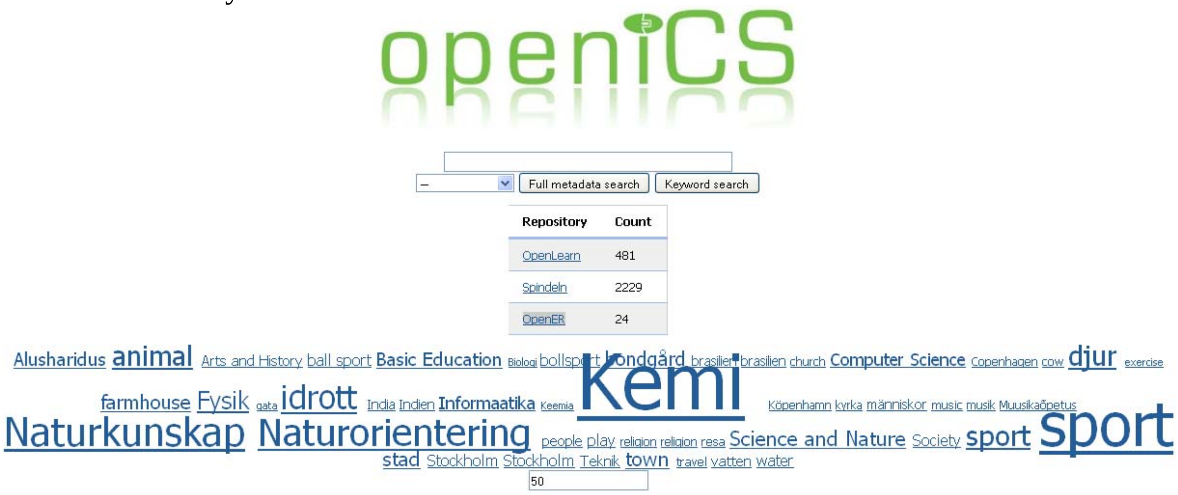

Figure 3: The UI of the OICS prototype

3.2.1 Keyword cloud

Keywords from all available metadata records (stored in the LOM fields “General.Keyword” and “Classification.Keyword”) are presented in a “cloud” where the size of each keyword corresponds to the number of records that contain it. Each keyword is hyperlinked to the search interface, pre-completed in a way where the two LOM keyword fields are queried for the keyword, and the search is automatically executed.

3.2.2 Repository list

All connected repositories are listed with the number of the objects they contain.

3.2.3 Search

The start page shows a simple text input field for entering search terms that can either be used in a full text search or limited to keywords in the same way as the keyword cloud. The full

text search uses the indexing service that is part of the OpenACS platform, where the complete metadata record is indexed.
An advanced search interface is also available, where the user can specify search criteria for any LOM field (see Figure 4).

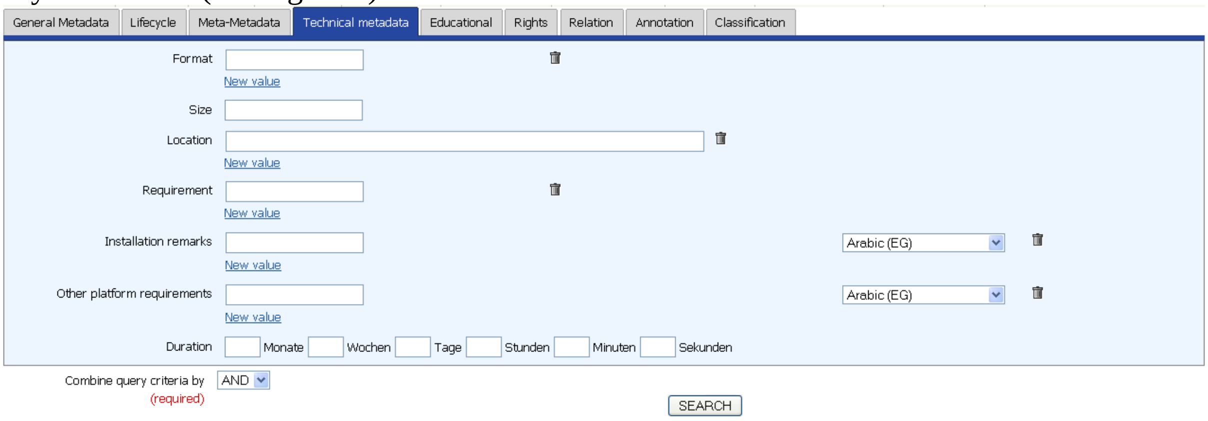

Figure 4: Advanced search interface (LOM)

All search criteria entered by the user can be combined either by the boolean AND or OR operators.

This form is not intended to be a model for the search interface of the final version of the OICS since it has become obvious from prior research (Najjar, 2006) that users are generally reluctant to work with large forms in order to find information. In this early phase of the project, this UI serves as a testbed for gathering requirements and evaluating efficiency of search algorithms. Whereas the full text search uses the tsearch2 indexing engine that is part of PostgreSQL, the advanced LOM search is translated into complex SQL queries.

4 Detailed technical capabilities

4.1 OAI-PMH harvester from KU Leuven

The ARIADNE Harvester (Ariadne, 2009) has been developed to manage harvesting of one or more OAI-PMH targets in a way that is automated as much as possible. The development has been mainly driven by European eContentplus projects which are using IEEE LOM formed metadata, and require validation against a certain application profile. Since no application profile has been defined yet for the OICS, the validation service of the harvester is currently only used for checking conformance to the LOM XML schema and to the vCard format.

4.2 SPI implementation via SOAP

SPI is a standardization effort currently undertaken by the CEN/ISSS Workshop on Learning Technologies aiming to produce a protocol for publishing learning objects and metadata to digital repositories (SPI, 2009). A SOAP binding of SPI is implemented both in the oaiharvester (it behaves as the source of the publication requests) and the new-lors repository (as its target).
Currently a very early version of the SPI specification is implemented and only one of the methods it provides (submitMetadataRecord, see Table 1) is used. K.U. Leuven has already

finished the implementation of the new version in its harvester tool, and the new-lors implementation is currently being updated.

Method name	**submitMetadataRecord**	
Return type	Void	
Parameters	*Name*	*Type*
	targetSessionID	String
	Identifier	String
	Metadata	String
Fault	NO_SUCH_SESSION INVALID_IDENTIFIER INSUFFICIENT_CREDENTIALS INVALID_METADATA	

Table 1: SPI method submitMetadataRecord

SPI currently uses the same mechanism for session management that has been defined for the SQI protocol but its implementation on the OICS currently does not require credentials as it is configured to only accept SPI requests on its loopback networking interface. Both the harvester and the repository are installed on the same machine, and SPI connection from the outside is currently disabled.

4.3 LOM implementation with some limitations

Some technicalities of LOM are not yet correctly handled by the 1st implementation and are currently being added to the metadata service:

- LOM defines some fields as being ordered. The current implementation does not yet store the order of values for these fields.
- LOM provides the possibility to use different vocabularies referenced by an identifier and store in the source field of the element. The current implementation only permits the storage of values defined in the standard LOM vocabularies for these fields.
- Translations of string values currently are not grouped, i.e. for a given metadata field, all strings are stored in a flat list.

5 Content repositories

The following repositories have provided content for the 1st generation of the OICS

Repository name	*Institution*	*Type*	*Quantity &definition*	*IPR*
The Spider, The Swedish learning repository federation	Umea university	Text, images, sound, movie, learning objects	2229 resources (no information about learning time currently encoded)	Mostly free resources
OpenER	Open university of the Netherlands	Self learning materials for High School / MBO level	750 hours (25 ECTS) in units of max. 25 hours (not yet encoded in metadata)	Creative Commons License 2.5 (bync-sa)
OpenLearn	Open university (UK)	Structured learning online courses	5970 hours according to metadata information	Creative Commons ShareAlike v2.0

6 Integration mini case studies

6.1 Use of OAI-PMH and SPI for metadata import

The possibility to harvest metadata from an OAI-PMH target and to publish it immediately into a repository provided by KU Leuven’s harvesting framework, proved to be an effective approach for this first generation of the OICS. It remains to be seen if this approach also delivers enough flexibility for the more sophisticated use cases, which will be implemented in the next generation.

6.2 Metadata import from an RSS feed

The OpenLearn repository does not yet provide an OAI-PMH target for harvesting metadata about its learning objects. As an alternative, an RSS feed exists at http://openlearn.open.ac.uk/rss/file.php/stdfeed/1/learningspace.xml from which identifiers of these objects can be extracted by parsing the link element of each item. A metadata file is available for each identifier, and can be fetched from an URL constructed from a given template (http://openlearn.open.ac.uk/rss/file.php/stdfeed/<*identifier*>/formats/metadata.xml). While practically feasible for the purpose of a protoype, this procedure will not stand up to the requirements of a production environment. The following problems were encountered:

1. Unreliable identifiers: The identifiers extracted from the RSS feed do not always match the identifier found in the metadata file.
2. Incremental updates: Contrary to OAI-PMH which permits selective harvesting of metadata records changed since the last harvesting, this procedure does not permit to distinguish between unchanged and updated records.
3. The metadata schema used by OpenLearn was not unambiguously mappable to LOM. For example, it has a *time* field, which has an integer value without indication of the unit.

6.3 LOM

The LOM schema currently maintained in the OICS prototype has not yet been profiled for a specific use case, since it has not yet been decided which use cases the next generation of the OICS will support.

7 Future directions

7.1 Development of Use Cases

Requirements engineering (RE) is a critical phase of the systems development life cycle. The aim of the RE process is to deliver a requirements specification document which defines the system to be developed. It is often difficult to understand requirements if they are stated as a list. Hence, functional requirements as well as non-functional requirements are expressed in **use cases.**

In the project preparation, a set of basic use cases have already been collected that somewhat illustrate how different work packages would deploy OICS. These are documented in the project's "Description of Work". In addition, in the beginning phase of the project, WP1 has invited project partners, especially the work package partners, to contribute more use cases, given that they have developed more insights into their respective goals, needs and tasks. We have developed, based on the literature of requirement engineering, a template (Appendix A) which partners should employ to describe their use cases. An overview of the use cases collected through this process is shown below:

Source: iCOPER Description of Work	Contributions from Partners
Modeling of Learning Outcomes and Opportunities (WP2)	Content Authoring
Instructional Modeling (WP3)	Discuss a learning object
Content Authoring (WP4)	Create a personal collection
Learning Delivery (WP5)	Four roles: student, tutor, librarian, ontology administrator
Assessment (WP6)	Personalised Search for Learning resources
Evaluation (WP6)	Recommending resources

Apparently and inevitably there are overlaps among these use cases. It is necessary to compile these use cases, and more challenging and important, is to convert this requirement information into specifications for the development team to implement. Furthermore, the requirements described therein may compete for the development resources of WP1. Consequently, ***prioritization*** of the requirements identified is deemed necessary. A decision-making process based on the three major criteria will be implemented:

- ***Feasibility***: an estimate of the requirement feasibility in terms of cost, impact and potential technical issues is attached to the requirement
- ***Importance***: indicates how important the requirement for users is. It can be assigned a nominal rating such as:
    - No need at all
    - No harm to have
    - Nice to have
    - Must have
- **Urgency**: indicates how urgent the requirement fulfillment for users is. It can be assigned an ordinal rating such as:
    - Very high
    - High

- Medium
- Low
- Very low

In addition, the following considerations need to be addressed as well:
Quality Criterion/Metric: The solution applied for fulfilling the requirement should be evaluated against the criteria, which preferably quantitative ones.
Dependencies: It is relevant to identify the dependencies among different requirements, because changes applied to one may have significant impacts on the dependent ones
Incompatibility: It is also relevant to identify whether the requirements are in conflict. Conflicts that are caused by mistake can be resolved in a relatively straightforward manner. Other conflicts are aroused because of true differences in opinion/intention. These are the conflicts that might eventually need to be addressed using negotiation techniques.
Nonetheless, collection of further requirements may be undertaken, given that the quality of requirements information never attains perfection. The point at which the requirements process terminates is therefore a matter of judgement about when and whether the collected requirements are a sufficiently sound basis for development to commence. Requirements activities continue to be enacted concurrently with down-stream activities as user requirements change, as design options have to be traced, and as errors and omissions in the requirements specification emerge.

7.2 Technical Challenges

The following technical problems have been identified in the discussions between work package partners. The current prototype does not yet deal with these problems, but state of the art solutions to them will be included into the next generation:

- Efficiency of search algorithms: The two approaches currently implemented in the OICS prototype (mapping interface query to SQL statements, and using a full text indexing engine) have to be evaluated.
- Identifying resources: What is the best way to guarantee that resources have persistent identifiers across multiple instances of metadata records that refer to them or even across multiple copies of the same resource?
- Access to content and integration with LMS: Which scenarios for content delivery does a learning object repository need to support?
- Identifying users: How can users be identified in a way that advanced services can be delivered to them across system boundaries?

8 References

OpenACS (2009). OpenACS, available at http://openacs.org/, consulted February 25th 2009.
DotLRN (2009). Product overview, available at http://dotlrn.org/product/overview/, consulted February 25th 2009.
Ariadne (2009). Harvesting LOM, http://ariadne.cs.kuleuven.be/lomi/index.php/Harvesting_LOM, consulted February 25th 2009
Ternier, S. et al. (2008). A Simple Publishing Interface For Learning Object Repositories. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008* (pp. 1840-1845). Chesapeake, VA: AACE.
Najjar, J. (2008). *Learning Object Metadata: An empirical investigation and lessons learned*. PhD Thesis. http://www.cs.kuleuven.be/publicaties/doctoraten/cw/CW2008_03.abs.html.
LOM (2009). Draft Standard for Learning Object Metadata, available at http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, consulted February 25th 2009.
SPI (2009). SPI, available at http://ariadne.cs.kuleuven.be/lomi/index.php/SPI, consulted February 25th 2009.
OAI (2009). The Open Archives Initiative Protocol for Metadata Harvesting, version 2.0, available at http://www.openarchives.org/OAI/openarchivesprotocol.html, consulted February 25th 2009.
Bildungspool (2009). das eLearning-Portal des Bundesministeriums für Unterricht, Kunst und Kultur, available at http://bildungspool.bildung.at/, consulted February 25th 2009.
EducaNext (2009). EducaNext, available at http://www.educanext.org/, consulted February 25th 2009.
Tsearch2 (2009). Tsearch2 - full text extension for PostgreSQL, consulted February 25th 2009.

Appendix A: Use Case Template

Attribute	Definition	Example
Use Case	*Use case identifier* A unique name suggesting its purpose. The name should express what happens when the use case is performed and be an active phrase.	
Goal & Sources	*The main goal to be achieved by use case and* *Sources for the requirement*	
Actors	*List of actors involved in use case*	
Assumptions/ Conditions	*Conditions that must be true for use case to terminate successfully* List all the assumptions necessary for the goal of the use case to be achieved successfully. Each assumption should be stated as a statement that evaluates to be true or false. Use case ***extensions*** can be used to specify behavior when an assumption is false.	
Steps	*Interactions between actors and system that are necessary to achieve goal* The sequence of interactions necessary to successfully meet the goal. A step has the form <sequence number><interaction>	
Results	*Specify whether the goal delivery is successful or fails*	
Variations	*(optional) Any variations in the steps of a use case* In the form <step reference> < list of variations separated by **or**>	
Non-Functional	*List any non-functional requirements that the use case must meet.* In the form: <**keyword**>: < requirement>	
Issues	*List of issues that remain to be resolved* Some notes on possible implementation strategies or impact on other use cases.	
Use Case Extension	<extension identifier> **extends** <use case identifier>	
Change	*Goal to be achieved by extension*	

	This section documents the variant of the use case in terms of the assumption that discharges.	
Condition	*The condition that must be satisfied if the extension is to take place*	
Steps	*Changes to use case steps.* The changes are expressed in terms of new or altered steps that apply to a use case at an *extension point* (i.e. a reference to a step) if some condition is true. <step reference> <changes to step>	
Variations	*(optional) …*	
Non-Functional		
Issues		