

LUISA

*Learning Content Management System Using Innovative Semantic Web
Services Architecture*

IST- FP6 - 027149

Deliverable D4.8

LOMR reference implementation report and full documentation

Miguel-Angel Sicilia
Salvador Sanchez
Elena García Barriocanal
Elena Mena Garcés
Raquel Rebollo Fernández
Ramiro Cano Gómez
Alberto Abián Belmonte
Elisa Rodríguez Ruiz
Leonardo Lezcano Matías
Sinuhé Arroyo
Nikolaos Korfiatis

Due date of deliverable: 31/08/2008

Actual submission date: 06/10/2008

Start date of the project: 01/03/2006

Duration: 30 Months

Miguel-Angel Sicilia
University of Alcalá

Version 1.0, dated 30/09/2008

Change History

| Version | Date | Status | Author (Partner) | Description |
|---------|------------|---------------|---|---|
| 0.1 | 10.08.2007 | Initial Draft | Miguel-Angel Sicilia. | First draft |
| 0.2 | 19.08.2008 | Draft | Raquel Rebollo, Ramiro Cano Gómez, Elisa Rodríguez, Leonardo Lezcano, Elena Mena. | Contribution to section 5 |
| 0.3 | 21.08.2008 | Draft | Elena Mena. | Minor changes in section 2 |
| 0.4 | 27.08.2008 | Draft | Alberto Abián, Sinuhé Arroyo, Nikolaos Korfiatis. | Contribution to section 4 |
| 0.5 | 10.09.2008 | Draft | Salvador Sanchez, Elena García. | Contribution to section 5 |
| 0.6 | 16.09.2008 | Draft | Elena Mena. | Contribution to sections 3 |
| 0.7 | 20.09.2008 | Draft | Elena Mena. | Contribution to sections 4. |
| 0.8 | 22.09.2008 | Draft | Elena Mena. | Minor changes in section 1 and Executive summary. |
| 0.9 | 25.09.2008 | Draft | Elena Mena. | Annex added. |
| 1.0 | 30.09.2008 | Final | Miguel-Angel Sicilia. | Final version after review |



EXECUTIVE SUMMARY

The LUISA LOMR is an open-source software framework that provides the functions to store and query for learning objects stored in semantic form. The semantics are provided by using WSML stored in RDF format as the underlying data management mechanism.

The architectural prototype of the LUISA LOMR, described in the previous deliverable 4.3 served the role of an early assessment of the key architectural issues behind the LOMR. The present deliverable reports on the evaluation of the final reference implementation of the framework and other additional components.

Document Information

| | | | |
|---------------------------|--|----------------|-------|
| IST Project Number | FP6 – 027149 | Acronym | LUISA |
| Full title | Learning Content Management System Using Innovative Semantic Web Services Architecture | | |
| Project URL | http://www.luisa-project.eu / | | |
| Document URL | | | |
| EU Project officer | Francesco Barbato | | |

| | | | | |
|---------------------|---------------|-----|--------------|--|
| Deliverable | Number | 4.8 | Title | LOMR reference implementation evaluation report and full documentation |
| Work package | Number | 4 | Title | Learning Object Metadata Repositories and LOMR Integration Development |

| | | | | |
|-------------------------------------|---|------------|---|------------|
| Date of delivery | Contractual | 31/08/2008 | Actual | 30/09/2008 |
| Status | Version 1.0, dated 30/09/2008 | | final <input checked="" type="checkbox"/> | |
| Nature | Report <input checked="" type="checkbox"/> Demonstrator <input type="checkbox"/> Other <input type="checkbox"/> | | | |
| Dissemination Level | Public <input checked="" type="checkbox"/> Consortium <input type="checkbox"/> | | | |
| Abstract (for dissemination) | This deliverable describes the implementatin evaluation of the LUISA LOMR repository. | | | |
| Keywords | Learning objects, learning object repositories, Semantic Web, Semantic Web Services, ontologies, WSMO, IRS-III, IEEE LOM, IMS LD. | | | |

| | | | | |
|---------------------------|---|-----|--------------|--|
| Authors (Partner) | Miguel-Angel Sicilia (UAH), Salvador Sánchez-Alonso (UAH), Elena García-Barriocanal (UAH) | | | |
| Responsible Author | Miguel Angel Sicilia | | Email | msicilia@uah.es |
| | Partner | UAH | Phone | +34 91 886 6603 |

Project Consortium Information


| Partner | Acronym | Contact |
|----------------------------------|---|---|
| Atos Origin S.A.E. (Coordinator) | ATOS  | Nuria de Lama Atos Origin S.A.E. c/ Albasanz 12 E-28037 Madrid, Spain Email: nuria.delama@atosorigin.com Tel.: +34 91 214 9321 Fax:+34 91 754 3252 |
| University of Alcalá de Henares | UAH  | Dr. Miguel-Angel Sicilia Information Research Unit University of Alcalá Ctra. De Barcelona, Km 33.6 E-28871Alcalá de Henares (Madrid), Spain Email: msicilia@uah.es Tel.: +34 91 886 6603 Fax: +34 91 885 6646 |
| University of Uppsala | ULL  | Dr. Ambjorn Naeve University of Uppsala Kyrkogårdsgatan 2 C Uppsala Email: amb@nada.kth.se Fax: +46 184-716-294 |
| Open University | OU  | Dr. John Domingue Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom Email: j.b.domingue@open.ac.uk Tel.: +44 1908 655014 Fax: +44 1908-653-169 |
| University Henri Poincaré | UHP  | Dr. Monique Grandbastien University Henri Poincaré Vandoeuvre les Nancy 54506, PO Box 239, France. Email: monique.grandbastien@loria.fr Fax: +33 383-278-319 |
| Giunti Labs S.r.l. | GIUNTI  | Dr. Fabrizio Girogini Giunti Labs S.r.l. Abbazia dell'Annunziata Via Portobello Baia del Silenzio 16039 Sestri Levante (GE), Italy Tel.: +39.0185.42123 Fax: +39.0185.43347 |
| EADS FRANCE – Innovation works |  | Anne Monceaux EADS FRANCE – Innovation works Avenue Didier Daurat - Centreda 1, Toulouse, 31700, France. Email: anne.monceaux@airbus.com Tel.: +33 561-184-725 Fax: +33 561-187-611 |



TABLE OF CONTENTS

| | |
|---|-----------|
| EXECUTIVE SUMMARY | 3 |
| TABLE OF CONTENTS | 6 |
| 1 INTRODUCTION | 7 |
| 2 BACKGROUND: OVERALL ARCHITECTURE OF LUISA AND LUISA LOMR | 8 |
| 3 AVAILABLE OUTCOMES | 10 |
| 4 EVALUATION OUTCOMES | 11 |
| 4.1 Key artefacts and degree of coverage of planned use cases | 11 |
| 4.2 Chidamber & Kemerer metrics..... | 11 |
| 4.2.1 Weighted Method Per Class (WMC) | 11 |
| 4.2.2 Depth of Inheritance Tree (DIT) | 12 |
| 4.2.3 Number Of Children (NOC)..... | 12 |
| 4.2.4 Coupling Between Object classes (CBO)..... | 13 |
| 4.2.5 Response For a Class (RFC) | 14 |
| 4.2.6 Lack of Cohesion Of Methods (LCOM) | 14 |
| 4.3 Status of testing | 15 |
| 4.4 Performance conclusions..... | 22 |
| 5 ADDITIONAL DEMONSTRATORS | 24 |
| 5.1 Instructional design queries | 24 |
| 5.2 Learning resources and localized tourism..... | 25 |
| 5.3 Navigational queries | 27 |
| 5.4 Integrating ontologies and folksonomies..... | 29 |
| 5.5 OKI Repository | 31 |
| 6 REFERENCES | 34 |
| ANNEX 1: CODE ANALYSIS RESULTS | 35 |



1 INTRODUCTION

The mission of LUISA is that of exploiting the advantages of a Semantic Web Service Architecture to make richer and more flexible the processes of query and specification of learning needs in the context of Learning Management Systems and Learning Object Repositories.

This document describes the outcomes of the reference implementation. The intention of the work reported here is that of reporting on the evaluation of that implementation that exercises the main functionalities that are required in LUISA LOMR (Learning Object Metadata Repository), as defined in **D4.3 LUISA deliverable** labelled "*LOMR Architectural prototype specification*". These include the storage of learning object metadata in semantic form, the provision of a service-oriented interface and the import of data in non-semantic form, among others.

The document is structured as follows. Section 2 provides some background information on the LUISA LOMR and its positioning in the LUISA architecture. Then, Section 3 provides the information about the outcomes obtained in the final implementation. Section 4 reports on the results of the evaluations. Finally, Section 5 shows briefly a scope of each additional demonstrator developed using the LUISA LOMR core functionalities.

2 BACKGROUND: OVERALL ARCHITECTURE OF LUISA AND LUISA LOMR

The main components of the LUISA framework architecture are depicted in Figure 1.

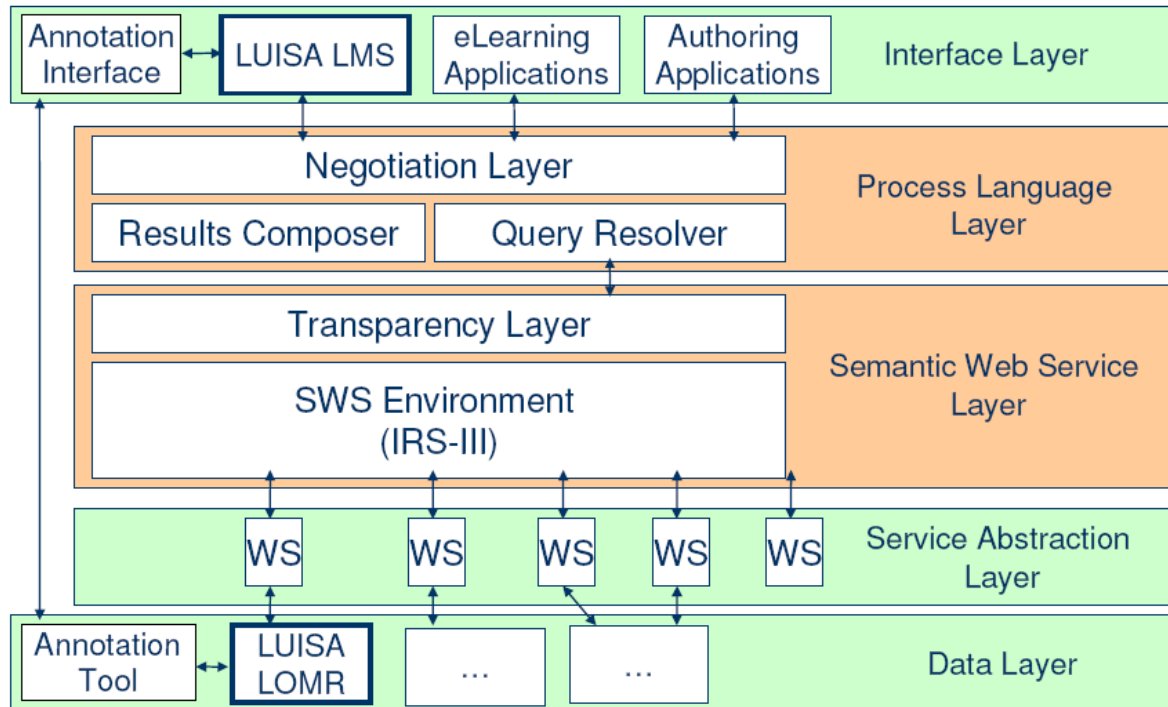


Figure 1 Main components of the LUISA architecture

The main architecture of the LOMR is divided in two layers as is depicted in following figure 2:

- *Triple (RDF) persistence layer.*
- *LUISA LOMR interfaces*

The *Triple persistence layer* abstracts applications and LOMR components from the RDF storage of WSML data. Eventually, applications might access the data at the RDF level, concretely the Annotation Tools are able to access the existing LOMRs through a Joseki server installation that provides the data at the RDF level.

The *LUISA LOMR interfaces* feature the interfaces to applications and the core components that are in charge of the queries and composition. The elements included in the implementation are described in **D4.3 LUISA deliverable**.

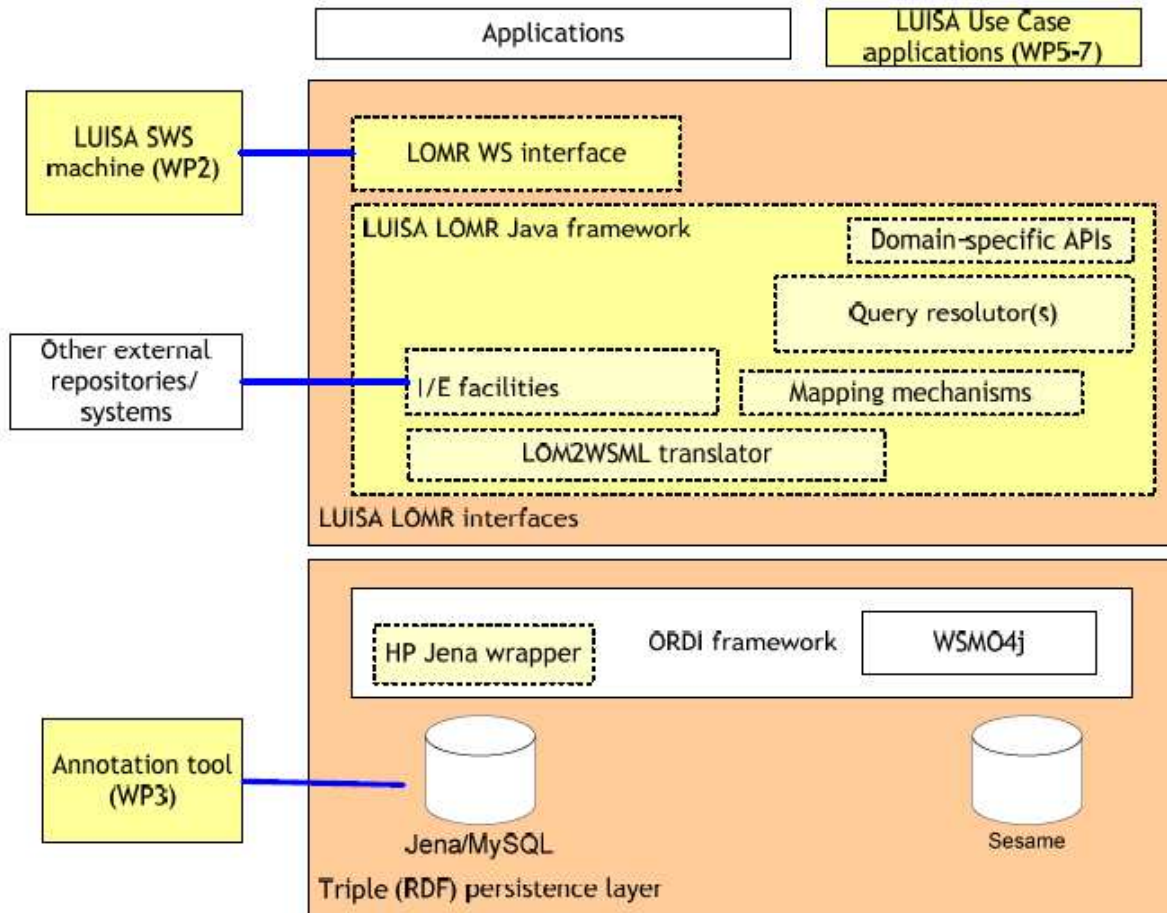


Figure 2 The LOMR framework architecture



3 AVAILABLE OUTCOMES

The core implementation of the LUISA LOMR and the additional components are available for downloading through the official LUISA website. In the zipped file you can find the following folder structure:

- luisa-lomr-v0.2: contains the source code and documentation of the luisa-lomr core, lom2wsml translator functionality, oki interfaces, navigational queries and ims-ld queries
- utils_luisa-lomr: contains the luisa-lomr-v0.2.jar file, the wsml ontologies and the config.xml file needed for the lom2wsml translator execution.
- luisa-lomrcompetencies-v0.2: contains the source code and documentation of the lomr+competencies implementation using the LOMR.
- common libraries: contains all the libraries needed for luisa-lomr-v0.2 and luisa-lomrcompetencies-v0.2 projects
- additional apps: contains the additional applications implemented that are not included in the previous projects mentioned.
- user manuals: contains the user manuals

For additional information about the implementation read the **D5.5 LUISA deliverable** labelled as “*Final reference implementation prototype*” or visit the Google code web pages of LUISA LOMR at: <http://code.google.com/p/luisa-lomr/>

4 EVALUATION OUTCOMES

4.1 Key artefacts and degree of coverage of planned use cases

The following Table summarizes the status of the key deliverables of the reference implementation.

| Deliverable | Description | Status |
|---------------------------------------|---|--|
| Repository code base | The code base for the evolving reference implementation. | Published and tested against the use cases identified. |
| User manual | Manual for API users of the LOMR. | Version developed. |
| Instances of LOMR for LUISA use cases | Instances of the LOMR to cover the academic and industrial use cases. | Developed and running. |

4.2 Chidamber & Kemerer metrics

Chidamber & Kemerer [3] software metrics suite calculate the reusability of classes, and the maintenance effort through the specialized metrics explained bellow, in the Annex 1 you can find the detailed results obtained.

4.2.1 Weighted Method Per Class (WMC)

Weighted Methods Per Class is defined as the sum of the complexities of all methods or the number of methods of a class. WMC is a predictor of how much time and effort is required to develop and maintain the class.

The results of the methods analyzed are mostly low (1-34), obtaining a maximum value at eu.luisa.lomr.oki.wsml.LOMROKISerializedLomImpl class.

Classes with many methods are likely to be more application specific, limiting the possibility of reuse. A large number of methods also means a greater potential impact on derived classes, since the derived classes inherit (some of) the methods of the base class.

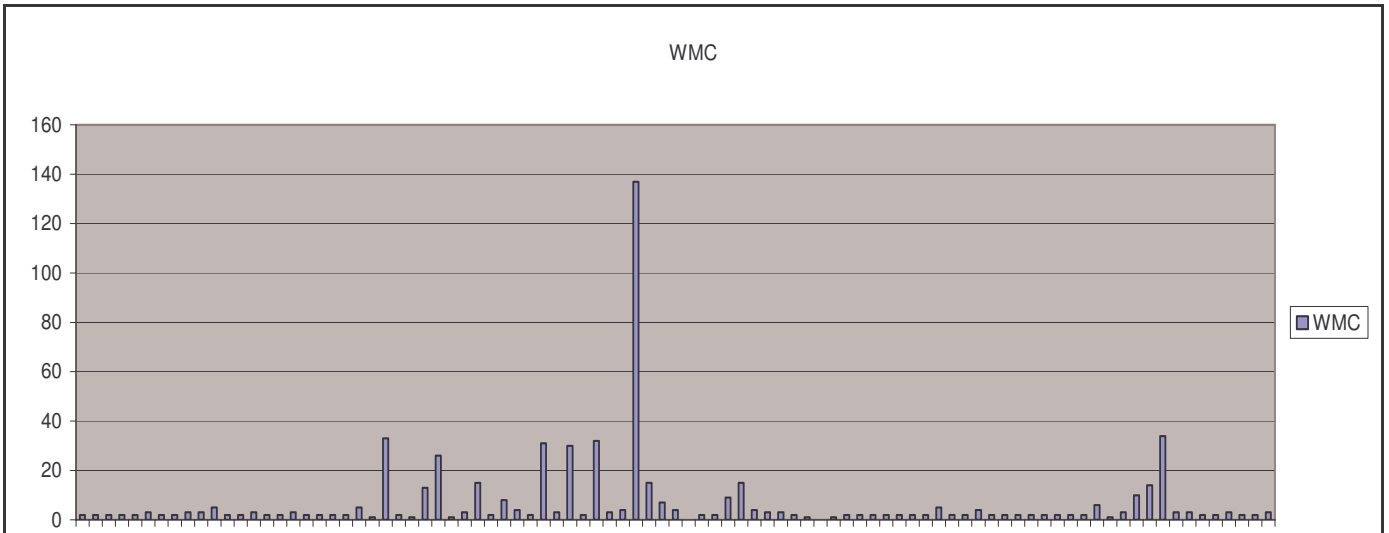


Figure 3 Weighted Method per Class (WMC)

4.2.2 Depth of Inheritance Tree (DIT)

The deeper a class is in the hierarchy, the more methods it is likely to inherit, making it more complex. Deep trees as such indicate greater design complexity. As a positive factor, deep trees promote reuse because of method inheritance.

A high DIT has been found to increase faults. A recommended DIT is 5 or less. The results of the analyzed classes are lower than the recommended DIT (1-3), what indicates that the time and effort required to maintain the classes is not high.

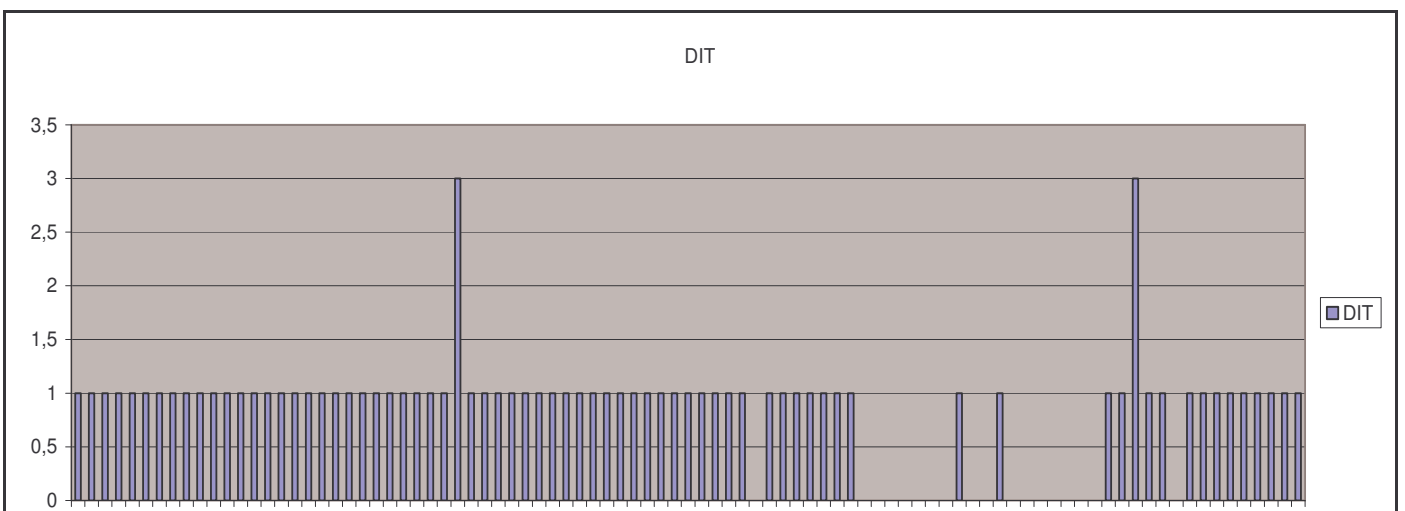


Figure 4 Depth of Inheritance Tree (DIT)

4.2.3 Number Of Children (NOC)

NOC is counted via the Inherits statement. It equals the number of immediate child classes derived from a class via the Inherits statement.

High NOC indicates high reuse, since inheritance is a form of reuse. A large number of children (high NOC) may also mean improper abstraction of the

parent class. If a class has too many children, it may indicate misuse of sub-classing. A class with many children may also require more testing.

The results of the analyzed methods are mostly 0, getting higher at classes where inheritance is needed.

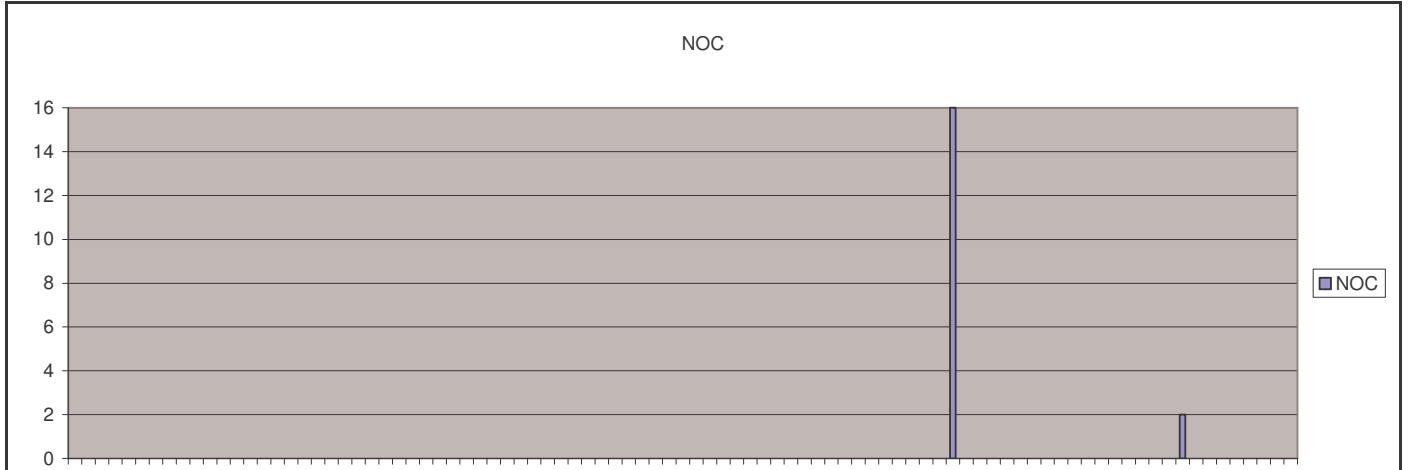


Figure 5 Number Of Children (NOC)

4.2.4 Coupling Between Object classes (CBO)

Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class. The uses relationship can go either way: both uses and used-by relationships are taken into account, but only once.

High CBO is undesirable. Excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application. In order to improve modularity and promote encapsulation, inter-object class couples should be kept to a minimum.

The CBO obtained are mostly lower than 40, what indicates that the coupling is correct. Some high values are found at example classes, what is not relevant for the maintenance.

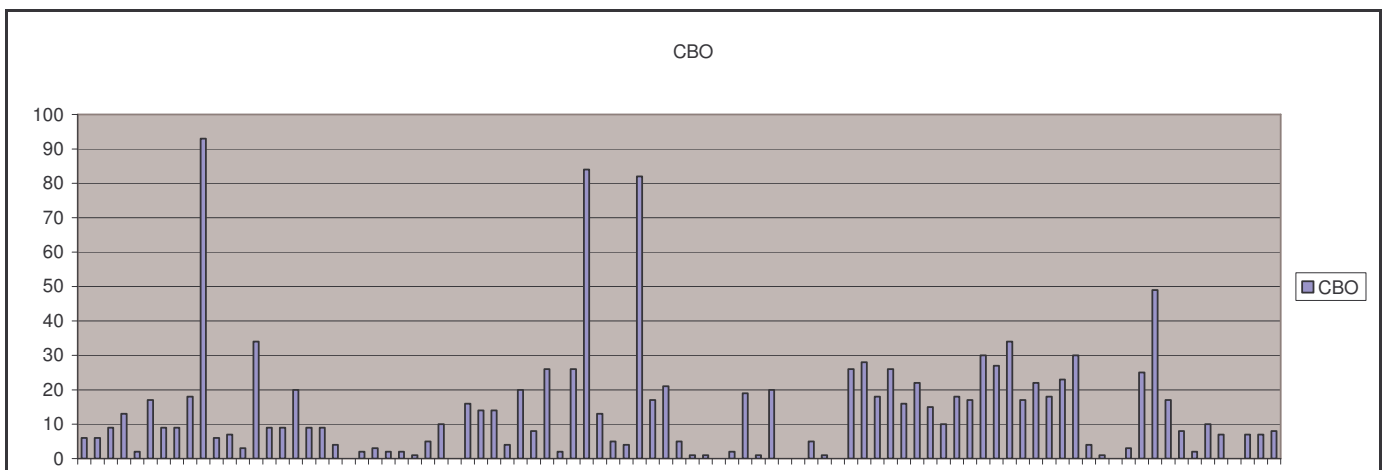


Figure 6 Coupling Between Object classes (CBO)

4.2.5 Response For a Class (RFC)

The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class. RFC is simply the number of methods in the set.

A large RFC has been found to indicate more faults. Classes with a high RFC are more complex and harder to understand. Testing and debugging is complicated. Most of the analyzed classes are low, so the complexity will not be affected.

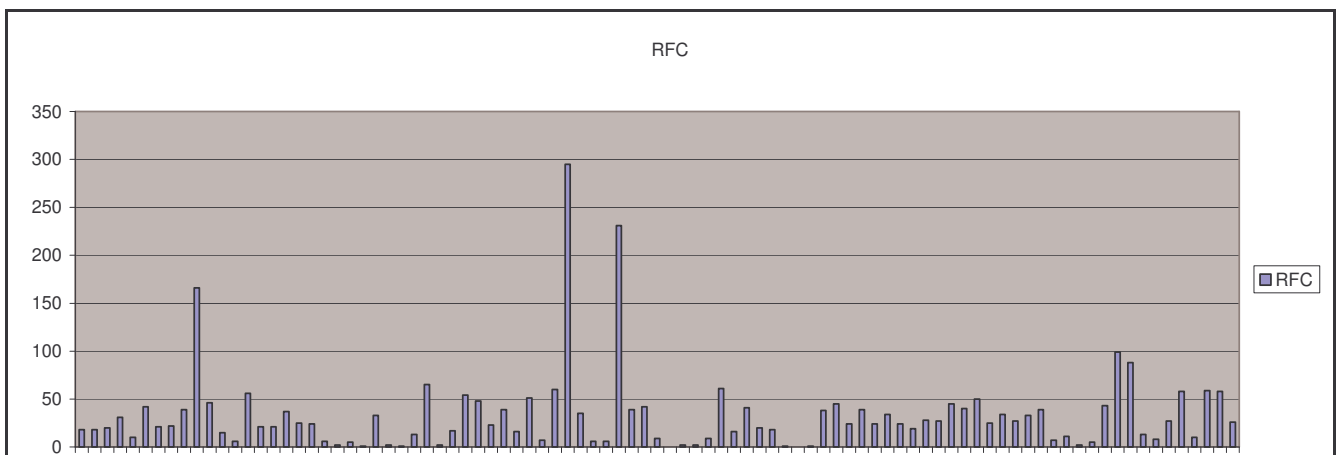


Figure 7 Response For a Class

4.2.6 Lack of Cohesion Of Methods (LCOM)

This metric can be used to identify classes that are attempting to achieve many different objectives, and consequently are likely to behave in less predictable ways than classes that have lower LCOM values.

After analysing the classes, the results are mostly low, except at “eu.luisa.lomr.oki.LOMROriginalLom” class that requires access to many attributes, in order to serialize a learning objet.

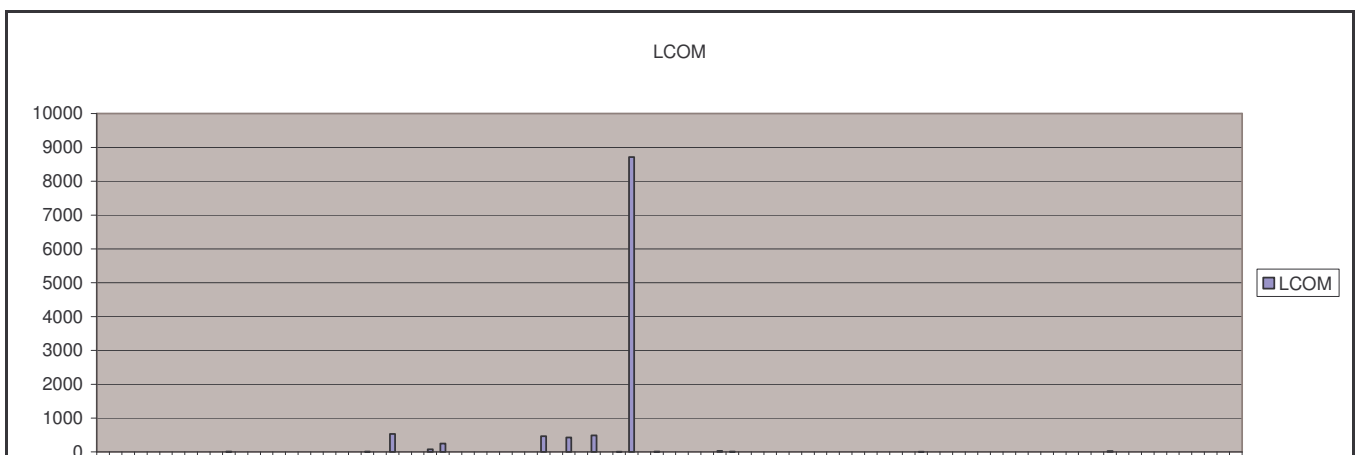


Figure 8 Lack of Cohesion Of Methods (LCOM)

4.3 Status of testing

Testing on the reference implementation has been done at the following levels:

- Unit testing using JUnit tests.
- A load test.

JUnit tests are spread across the different modules. The following table gives an overview of the distribution of unit tests across packages.

| Package | Number of classes | Number of test classes |
|-------------------|-------------------|------------------------|
| lomr | 90 | 1 |
| lomr+competencies | 108 (additional) | 27 (additional) |

The *load test* aimed at observing the effect of two variables in the response time of a server with an instance of the LOMR.

The storage test has been performed using an special example class in luisa-lomr distribution, `StoringFromFolderLoadTest` class. This class translates an user specified number of XML learning objects to an ontology representation of those learning objects in WSML, and stores them in a MySQL database.

The graph represents the time that takes the program to translate and store the LOs. The translation and storage of 10 learning objects takes 148 seconds meanwhile the translation and storage of 150 learning objects takes 22826 seconds. The storing time indicates the total time that the program requires to translate and store the number of the learning objects indicated.

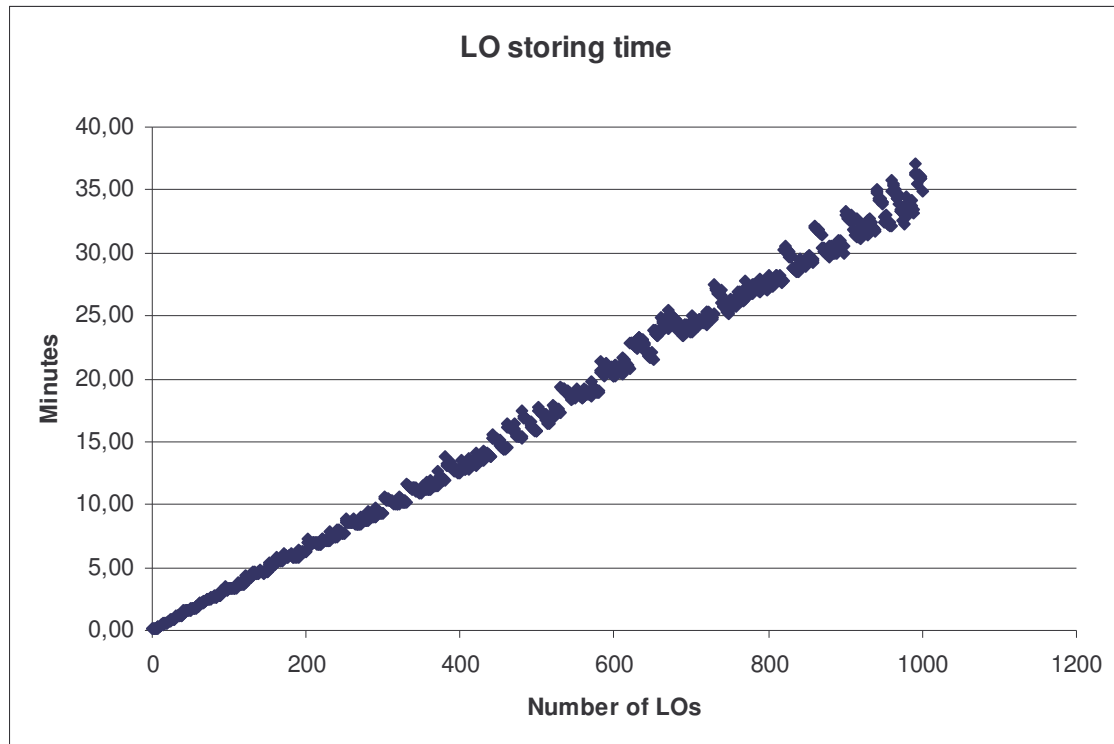


Figure 9 LO storing time

The WSML file with the initial ontology requires 90 KB of free HD (hard disk) space meanwhile the ontology stored in a database needs 1 MB of free HD space.

The xml files with 1000 Learning Objects require 7.82 MB of free HD space meanwhile the database whit 1000 Learning Objects translated and stored requires 115 MB of free HD space. So the average HD space required for each LO translated is 119 KB. There has been generated 85123 instances with the stored Learning Objects, the average HD space required for each instance is 97 bytes.

The MySQL database generated in the previous step must be loaded into main memory when the LOMR repository is opened. The time needed to open the repository with 20 LOs is 14 seconds meanwhile the same operation with 1000 LOs takes 821 seconds (approximately 13 minutes).

The computer used for the tests is an Intel Core 2 Xeon 1.6 GHz, 8 GB of RAM and Microsoft Windows Server 2003 R2 Enterprise Edition with Service Pack 2.

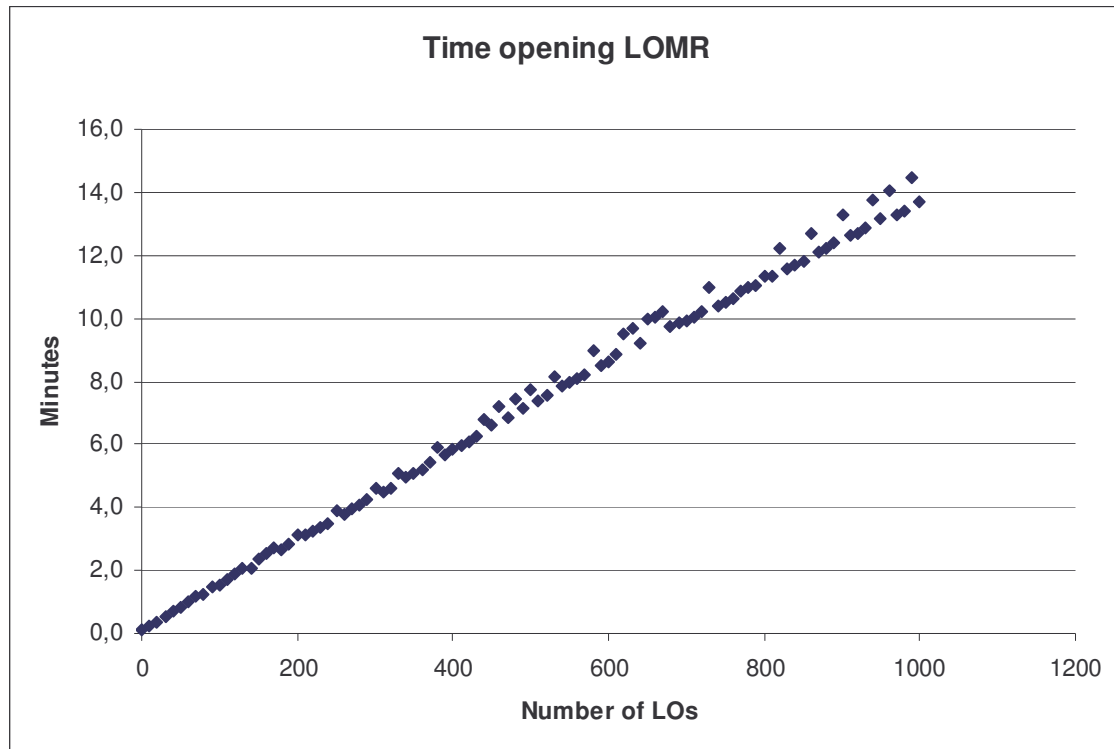


Figure 10 Time opening LOMR

The load test has been performed using the soapUI application that is a free and open source desktop application for inspecting, invoking, developing, simulating/mockng and functional/load/compliance testing of web services over HTTP.

Two repositories have been used in this test:

The UHP LOMR repository which requires 3.55 MB of HD space with 3739 instances.

The EADS LOMR repository which requires 13 MB of HD space with 2656 instances.

Performance for each Web Service:

- WSUhp:

Step 1 [getCompetenciesProvidedByLO]: took 143 ms

Step 2 [getCompetenciesProvidedByLOGroup]: took 17 ms

Step 3 [getCompetenciesRequiredByLO]: took 10 ms

Step 4 [getCompetenciesRequiredByLOGroup]: took 10 ms

Step 5 [uhpLOsearchByAuthor]: took 11 ms

Step 6 [uhpLOsearchByCompetency]: took 39 ms

Step 7 [uhpLOsearchByTopic]: took 11 ms

Step 8 [uhpLOsearchByTopicGroup]: took 13 ms



TestCase finished in 254 ms

- WSLomUhp

Step 1 [generateNewUri]: took 14 ms

Step 2 [getLOByStringUhp]: took 14 ms

Step 3 [getLOMetadata]: took 11 ms

Step 4 [getLOMhasPart]: took 10 ms

Step 5 [getLOMTiDesLeaCos]: took 9 ms

Step 6 [getLOMTiDesLeaForReCos]: took 10 ms

TestCase finished in 68 ms

- WSComposeUhp1

Step 1 [compose]: took 22 ms

Step 2 [delete]: took 14 ms

Step 3 [getManifest]: took 9 ms

TestCase finished in 45 ms

- WSComposeUhp

Test started at 2008-07-22 11:59:15.937

Step 1 [compose]: took 21 ms

Step 2 [composeWithAuthor]: took 18 ms

Step 3 [delete]: took 12 ms

Step 4 [getManifest]: took 10 ms

TestCase finished in 61 ms

- WSEads

Step 1 [eadsLOsearchByCode]: took 15 ms

Step 2 [eadsLOsearchByCompetency]: took 49 ms

Step 3 [eadsLOsearchByDomain]: took 9 ms

Step 4 [eadsLOsearchByField]: took 14 ms

Step 5 [eadsLOsearchByFunction]: took 12 ms

Step 6 [eadsLOsearchByKeyword]: took 15 ms

Step 7 [eadsLOsearchByProfession]: took 22 ms

Step 8 [eadsLOsearchBySpeciality]: took 13 ms

Step 9 [getCompetenciesProvidedByLO]: took 9 ms

Step 10 [getCompetenciesProvidedByLOGroup]: took 9 ms

Step 11 [getCompetenciesRequiredByLO]: took 8 ms

Step 12 [getCompetenciesRequiredByLOGroup]: took 9 ms

TestCase finished in 184 ms



- WSLomEads

Step 1 [generateNewUri]: took 10 ms

Step 2 [getLOByStringEads]: took 12 ms

Step 3 [getLOMetadata]: took 10 ms

Step 4 [getLOMhasPart]: took 9 ms

Step 5 [getLOMTiDesLeaCos]: took 8 ms

Step 6 [getLOMTiDesLeaForReCos]: took 10 ms

TestCase finished in 59 ms

- WSComposeEads1

Step 1 [compose]: took 18 ms

Step 2 [delete]: took 10 ms

Step 3 [getManifest]: took 16 ms

TestCase finished in 44 ms

- WSComposeEads

Step 1 [compose]: took 19 ms

Step 2 [delete]: took 9 ms

Step 3 [getManifest]: took 6 ms

TestCase finished in 34 ms

The previous times has been obtained using only one thread requesting each web service operation. These measures are not enough to know how the implementation will behave in a real environment, so additional tests are needed.

The next test simulates a specific number of users using the application for 180 seconds, response times and the number of bytes transmitted have been measured.

Every test is made up of a specific number of Java Threads that make requests at the most used Web Service operations with preconfigured parameters as the learning object language and the learning object identifier of existing instances in the LOMR repository. The most used Web Services operations are the following:

- UHP Web Service operations:

- getCompetenciesProvidedByLOGroup
 - getCompetenciesRequiredByLOGroup
 - uhpLOsearchByCompetency
 - uhpLOsearchByTopicGroup
 - getLOMetadata
 - getLOMhasPart

- EADS Web Service operations:

eadsLOsearchByCompetency
 getCompetenciesProvidedByLOGroup
 getCompetenciesRequiredByLOGroup
 getLOMetadata
 getLOMhasPart

The time used in all the test cases is 180 seconds and the number of Threads (it is assumed that each thread would be a user in the real environment) take the next values: 1, 25, 50, 75, 100 and 125. The computer used for the tests is an Intel Pentium D at 3.00GHz, 2.00 GB of RAM and Microsoft Windows Xp Pro. with SP2 installed. The following graphs can be used as a summary of the situation described above.

In 180 seconds many requests have been made, the next graph shows the shortest time of all de answers in UHP and EADS LOMR repositories:

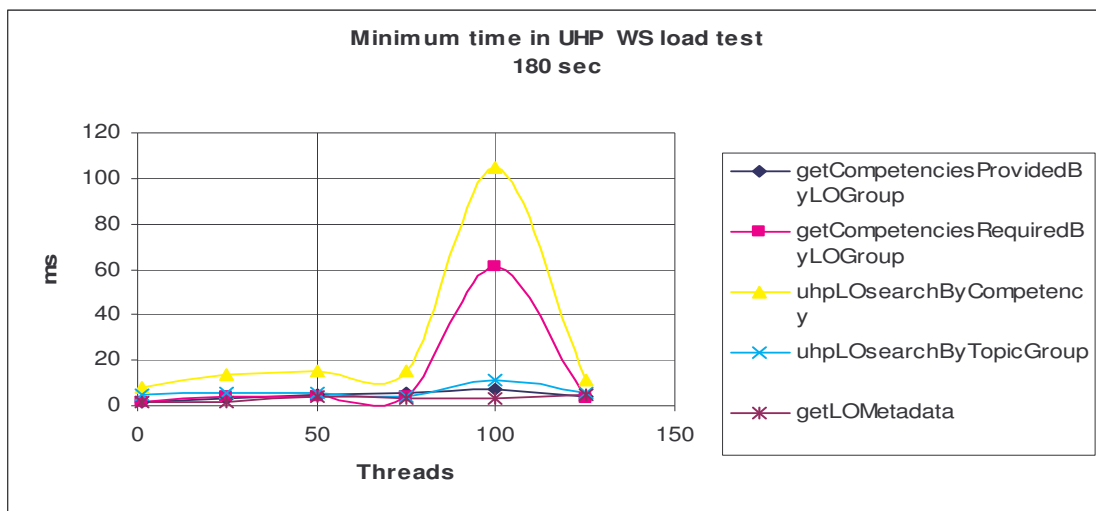


Figure 11 Minimum time in UHP WS load test

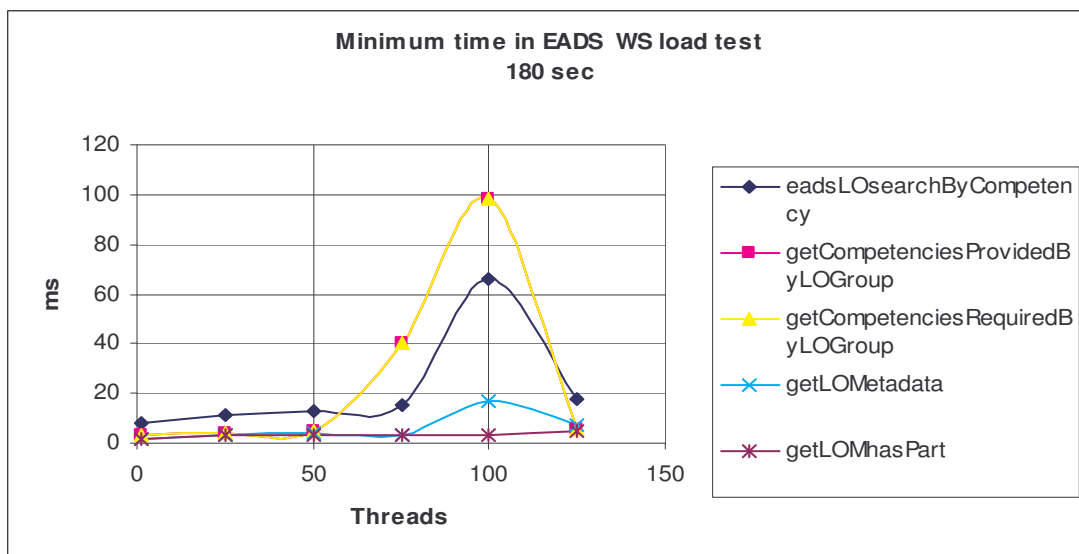


Figure 12 Minimum time in EADS WS load test

The two graphs below show the longest time in obtaining a response from the repository:

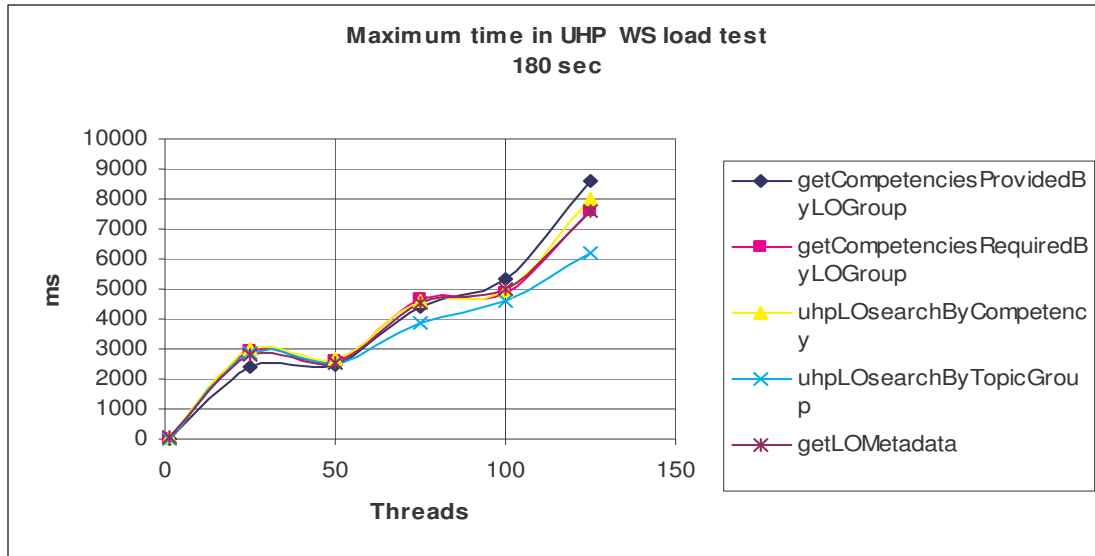


Figure 13 Maximum time in UHP WS load test

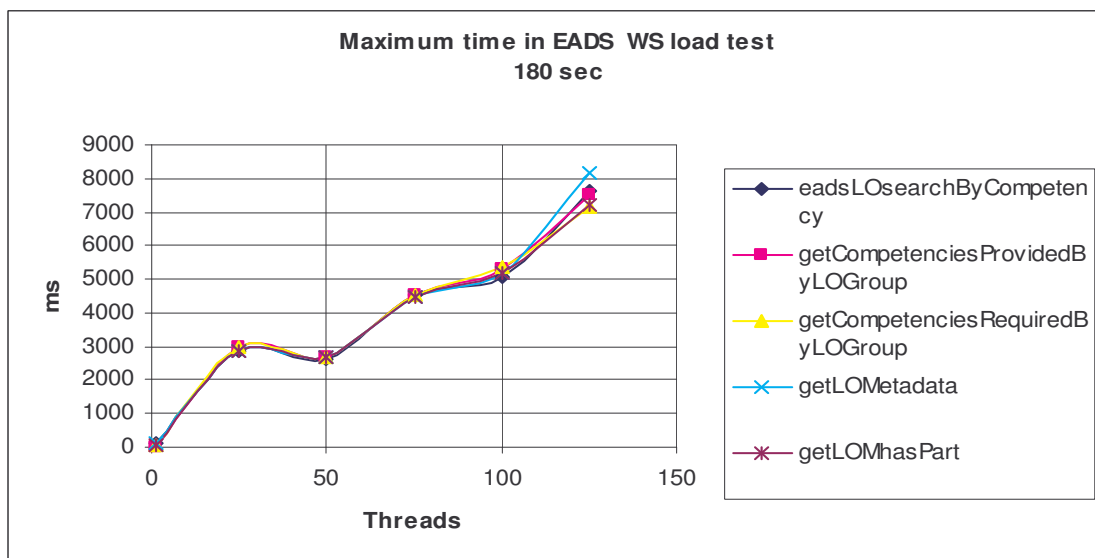


Figure 14 Maximum time in Eads WS load test

By performing the tests, it has been generated a large amount of bytes that must be transferred between the server and the client. Depending on the connection speed, it can become a very important factor in terms of application performance. The following graphs show the number of bytes transmitted in the LOMR Repository of UHP and EADS.

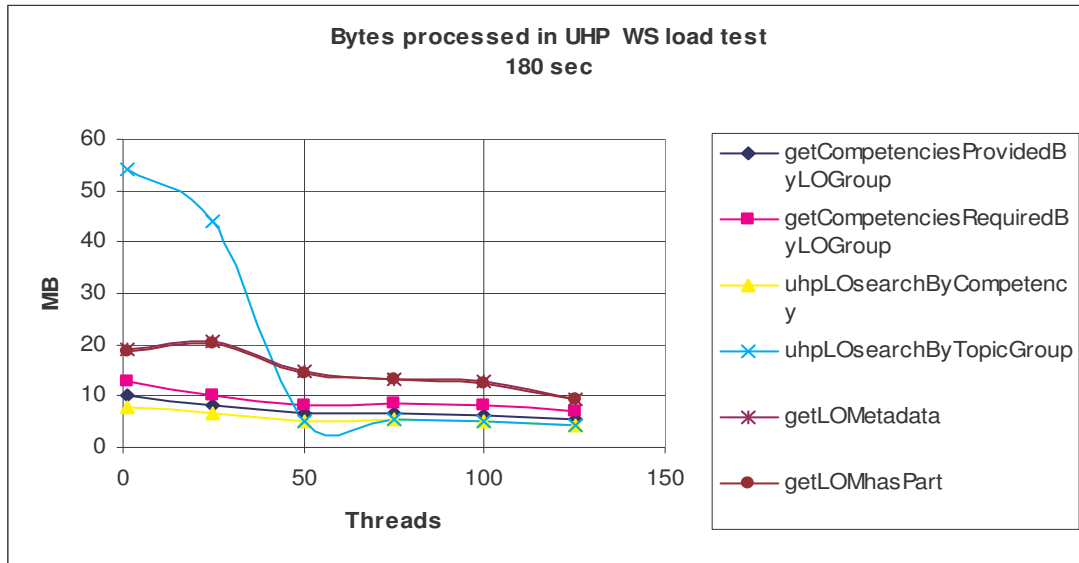


Figure 15 Bytes processed in UHP WS load test

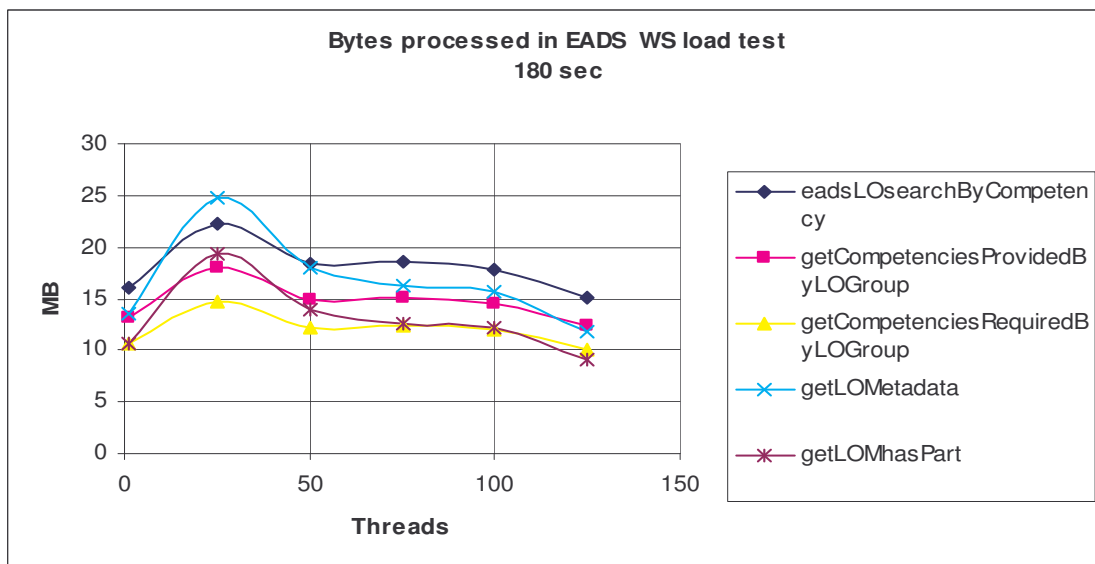


Figure 16 Bytes processed in EADS WS load test

4.4 Performance conclusions

As every software systems, the response time increases the more amount of data they store. The repository has to search in a larger collection of information or perform the modifications requested by the user in that collection. It is important to note that there are other underlying technologies used by the LOMR repository. The learning resources are stored ultimately in a relational database (MySQL), an application server (Tomcat) is used to host the repository, the semantic capabilities of the LOMR repository are achieved through the use of experimental libraries (Jena, ordi) and reasoners (mins). These experimental technologies are constantly evolving and changing. The operations carried out by the Operative System must be added also to this situation: reading or writing files, handling network connections...



Even as explained above, LOMR has a good performance when the number of users increases, rising to stabilize the used memory and response times of the tomcat server. The time used by de storage and initialization operations is not a problem because they are operations that are not performed very often and they are requested only by the administrator of the LOMR. Searching operations are requested very often by dozens of users. As shown in the graphics above, the times are reasonably shorts in the search processes.

5 ADDITIONAL DEMONSTRATORS

5.1 Instructional design queries

The IMS Learning Design specification is a meta-language that describes all the elements of the design of a teaching-learning process. Therefore, the IMS LD specification requires a modelling capable of describing explicitly and formally the semantics of its elements. To achieve this goal, we have developed an ontology which facilitates the semantic description of the A level conceptual model.

To develop the Learning Design ontology we have used a concept: taxonomy, designed by The University of Santiago de Compostela [1], which describes the elements of the IMS LD conceptual model and the IMS LD information model, which formally constraints the semantics of the concept taxonomy on the basis of the explanations formulated in natural language in both information and behavioural models. The model represents the integration of the learning design within IMS Content Package to obtain a so-called 'unit of learning'.

IMS Learning Design is being used with IMS Content Packages to model units of learning, in which contents are described in an XML document called 'the package manifest'.

For the construction of the ontology we have developed the `imslld2wsmlTranslator` for creating and editing it. This uses the XML manifest for creating the learning design ontology. Applying the translator to more than one manifest, we can obtain an ontology with several learning design.

Once we have generated the learning design ontology, it could be also stored in a repository, to launch queries taking into account any criteria of learning design data, and it will be carry out by the query manager. The Learning Design comes to be a subconcept of Learning Object, in order to this the query results comes to be Learning Objects instances. The process is going to be described in the following figure.

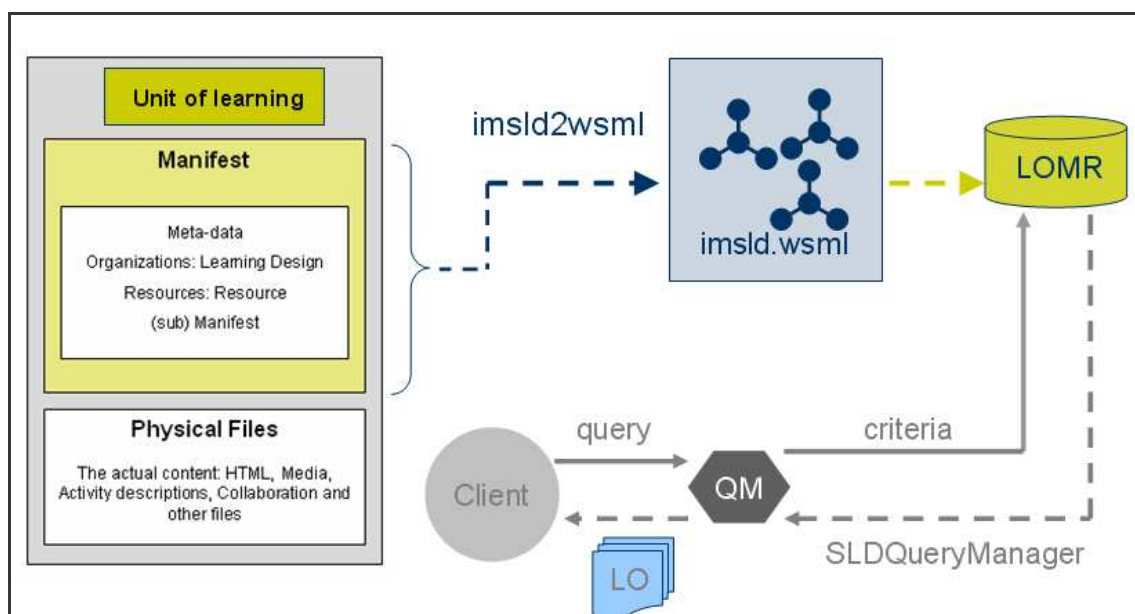


Figure 17 Learning Design implementation queries

One repository that contains the IMS Learning Design ontology has been created and has been developed according to LUISA project framework, we can check the compliance of learning object representations with concrete representations of instructional theories. This process assumes the availability of ontologies of instructional design [2] that follows the general structure of the "Theory One" ontology.

The query takes the ontology and executes some rules engine to check if any particular instance of the concept LearningObject fits with the constraints imposed by the methods of a given IDTheory (e.g. instructional design model).

The learning theory approach reported in this section complements the original LOMR approach with an emphasis on the inclusion of inference that allows to filter the resulting resources by learning theories criteria, which could be especially useful for applications used in academic contexts for the programming of learning activities personalized according to instructional design preferences.

5.2 Learning resources and localized tourism

This application explores the semantic capability in a Learning Resource storage and query application, providing a tour itinerary in a city. The result of the application will be a list of monuments which are interesting for the user, according to his/her inputs.

The information used by the application is historical personalities and monuments related to them through events that happened to that character in that monument. Places are also related to each other according to their distance. This information is shaped in a semantic way and stored in a database.

Once the information is stored, the user can make a query by establishing his/her artistic preferences, the historical personalities which interest him/her and the exact place where the user wants to start the visit.

The obtained results will be the monuments which the user is interested in ordered by different criteria.

The relationship between the concepts included in the ontology developed for this application is described in the following figure:

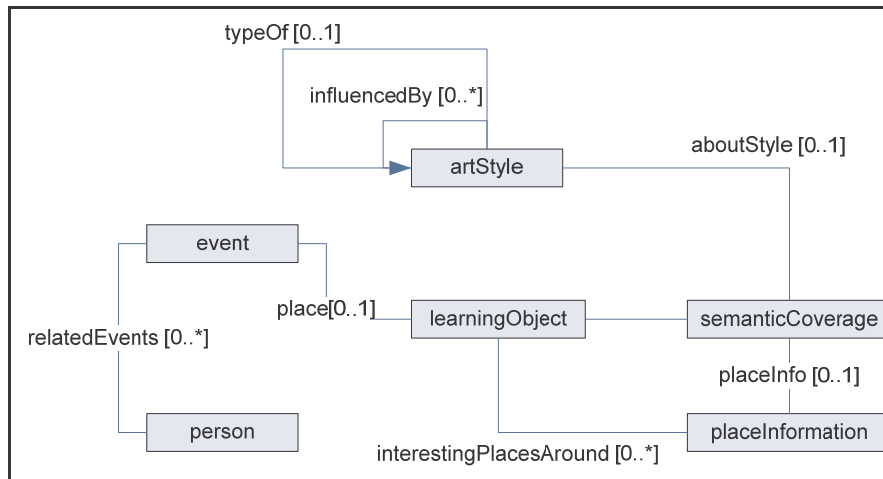


Figure 18 Art and architecture ontology relationship

The *event* concept saves events related to a historical character and to the place where they happened. E.g. Cervantes was_born in Cervantes' Birthplace Museum.

The *artStyle* concept saves artistic styles and subtypes with their influences.

The *person* concept saves historical personalities.

The *learningObject* concept saves the monuments. Inside learning object concept coverage attribute is found which saves the period, style and information related to the places where the monument is located.

The *placeInformation* concept saves the places near the monument classified in nearness rates: 0.1 km, 0.2 km, ...

The ontology will be fulfilled with the information required using instances of the previous described concepts. This semantic information will be stored in a dedicated database.

The user inputs will be the initial place where the visit starts, a list of characters and a list of artistic styles. A monument will be included in the final list if it fulfils the following conditions:

- That the monument style is:
 - One of the input styles list or,
 - A style influenced by one of the input styles list or,
 - A type of one of the input styles list.
- That in the monument an event related to one of the characters of the input characters list took place.

The final list will be ordered through nearness to the initial place. If some monuments of that final list have the same distance to the initial place, and are related to the same character, the monuments will be ordered through their date.

If the user selects one or more personalities without selecting the artistic style, the result will be a list of places in which there are events related to the

historical personalities. Results are ordered, first of all, through nearness to the initial place, and secondly, chronologically.

Supposing the user selects personalities and artistic styles, the result will be a list of places in which there are events related to the historical characters, according to the chosen styles. Results are ordered through nearness to the initial place, and chronologically rearranged provided that places are related to the same character and the distance between them is lower than 0.1 km.

Example:

User input:

Initial place: Santos Niños' Square

Styles list: Gothic, plateresque and baroque

List of characters: Miguel de Cervantes, Calderón de la Barca, María de Guzmán, Melchor de Jovellanos.

Output:

1. Santos Niños' Cathedral (0.1 km)
2. Antezana's Hospital (0.2km)
3. San Ildefonso's Façade (0.5km, event related to Cisneros in 1478)
4. Teologos' School (0.5km, event related to Cisneros in 1497)
5. Manriques' College (0.5km, event related to Calderon in 1614)

This application has many uses. A town hall could use it to simplify tourists the visit of an unknown city, providing special machines with access to the application. On the other hand, a travel agency could integrate it in its web page to help visitors decide their destination according to their interests. In addition, in an educative context, students will be able to relate places to characters and analyze the predominant artistic styles in a city.

5.3 Navigational queries

The navigational queries system can be used for performing interactive searching sessions, by the navigation of ontology elements and its relations, in a totally ontology-independent process.

From the functionality perspective, the navigational process involves two elements: the refining process and the searching process. The refining process allows navigating through the elements in the ontology, by its specialization or side-relations. The searching process matches the criteria selected by the user against the resources stored in the semantic repository.

The navigational session is based on the existence of "interest points", an arbitrary set of any element in the domain-specific ontology where the user has focused his attention. The initial "interest points" for the navigation will be the "entry points" defined by the user in the configuration file.

These “interest points” will be refined during the session, by the selection of more specific elements in the ontology (sub-elements), or the addition of new elements related to the current “interest points”.

The searching process is carried out by the “query-by-example” standard LOMR method, and will match the user selections against the repository available resources.

Also, the navigational process integrates the use of automatic reasoning capabilities, in order to offer additional information to the user, based on the provided criteria.

From the architectural perspective, the navigational process involves two elements: the repository layer and the user interface layer. The repository layer is part of the LOMR system, comprising the repository itself, the navigational session manager and the navigational query manager. The session manager handles the interest points, whilst the query manager handles the searching queries launched against the repository.

The user interface layer is not integrated in the LOMR system, since many different interfaces can be implemented to use the navigational queries system. However, the proposed web-based user interface example (LOMR-nav project) can be taken as a reference implementation for further work. In this example interface, the communication with the session and query managers from LOMR is carried out by a session handler, which translates the web-based user interactions to the repository code interfaces.

The overall architecture of the navigational process is described in the following figure.

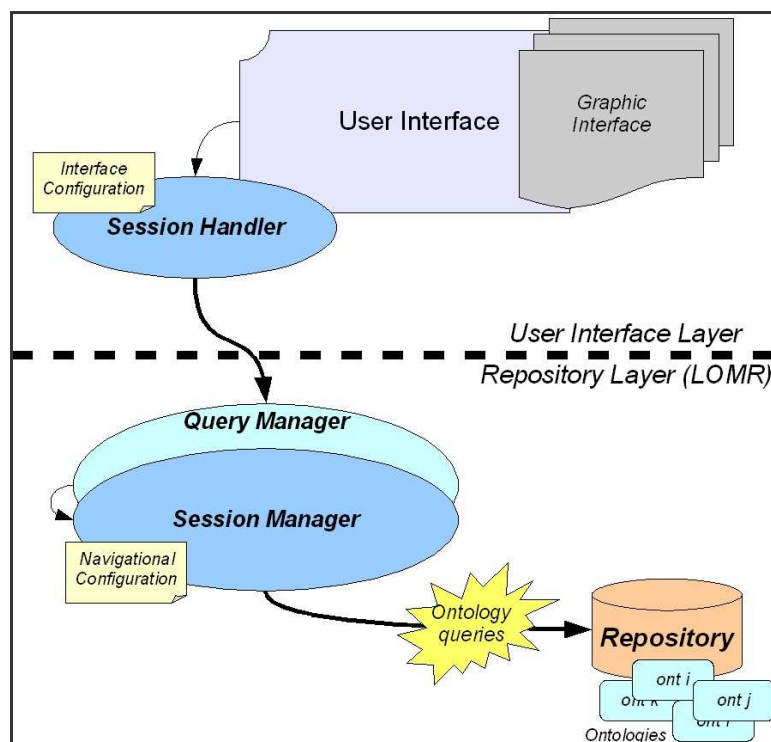


Figure 19 Navigational process



The navigational system configuration is read from an XML file containing the following structure.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<luisa>
  <entryPoint>
    <concept>
      <value>any_concept</value>
    </concept>
    <concept>
      <value>another_concept</value>
    </concept>
    <instance>
      <value>any_instance</value>
    </instance>
    <instance>
      <value>another_instance</value>
    </instance>
    <attribute>
      <value>any_attribute=value</value>
    </attribute>
    <attribute>
      <value>another_attribute=value</value>
    </attribute>
  </entryPoint>
</luisa>
```

This configuration sets the entry points used in the navigational process.

The proposed web-based user interface example also uses an independent configuration, which is read from an XML file containing the following structure.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<lomr-nav>
  <ontologyFile>path_to_the_file</ontologyFile>
  <ontologyConfigurationFile>path_to_the_file</ontologyConfigurationFile>
  <reasoningOntologyFile>path_to_the_file</reasoningOntologyFile>
  <reasoningQuery>reasoning_query_used</reasoningQuery>
</lomr-nav>
```

This configuration sets the domain-specific ontology used in the navigational process, the domain-specific ontology configuration file (which will contain the above-mentioned “entry points” configuration), the reasoning ontology file, and the reasoning query used by the reasoner to infer new relations.

The high interaction capabilities of this system make it ideal for user interfaces in non-strict search systems. For example, a student using a CMS could navigate through different terms related to a specific field, refining them to suit his preferences. When the student interests are captured in the navigational session, he would be able to search any matching learning resources in the repositories.

5.4 Integrating ontologies and folksonomies

This application called TagExplorer is an approach to integrate folksonomies and ontologies. TagExplorer is a plug-in for Mozilla Firefox that improves user experience when surfing across *del.icio.us* pages. It focuses on offering access to tags and content related to the one displayed on *del.icio.us* interface. For example, if you are visiting a tag named “cpu” the TagExplorer may give you some *del.icio.us* tag links like “transistor”, “cache”, “computer”, “amd”, “intel” or “dual-core” for you to click on and consult the bookmarks assigned to them. At



at the present, del.icio.us' interface already supplies a list of tags that shares some bookmark with the visited one. On the same way, TagExplorer is meant to find connected tags but based on its meaning (according to the knowledge represented inside the *OpenCyc* -<http://www.opencyc.org/>- ontology) instead of the shared bookmarks.

The Cyc knowledge base (KB) is a formalized representation of a vast quantity of fundamental human knowledge: facts, rules of thumb, and heuristics for reasoning about the objects and events of everyday life. The name "Cyc" (from "encyclopedia") is a registered trademark owned by Cycorp. Unfortunately the original knowledge base is proprietary, so this project uses a smaller version, created to establish a common vocabulary for automatic reasoning, released as *OpenCyc* under an open source license. Even though OpenCyc is less powerful than the original version, it perfectly fulfils the requirements of this project.

So far, delicious guiding aids include only tag clouds and lists of popular and related tags, which are dispersed over a non-hierarchical keyword categorization. By integrating the delicious folksonomy with the Cyc ontology we provide hierarchical relations between tags and we address a sort of problems which are intrinsic to folksonomy tagging like *plurals*, *polysemy*, *synonymy*, and depth (*specificity*) of tagging.

Furthermore, this will clearly complement the user possibilities of reaching desired pages because he will be able to exploit the knowledge in the ontology. As you can see in the "cpu" example, "transistor" and "cache" are CPU **parts** while "dual-core" is a **particular category** and "amd" and "intel" are CPU **brands**. Maybe those tags don't share any bookmark among them but they are semantically related and so can be recommended by the TagExplorer interface.

From the technical perspective, on the client-side some browser plug-in like GreaseMonkey for Firefox should be installed because our interface requires integration with del.icio.us incoming HTML code so we have to break the standard JavaScript security patterns, obviously for a good reason. The result is a frame embedded on the del.icio.us original web page containing the abovementioned tag links.

TagExplorer also needs an OpenCyc server from where it retrieves all the necessary information. The server includes an inference engine, a knowledge base browser and a Java API for writing programs that access and use the OpenCyc knowledge base. Of course, there is some information, like tag's URLs that has to be supplied to OpenCyc for allowing our application to transform related words found into del.icio.us tag links destined to redirect client to pages representing those tags. So, in general terms, TagExplorer functioning can be summarized as follows:

1. When a del.icio.us' client loads any tag then a GreaseMonkey script (`deliciousfull_url.user`) sends a GET request to TagExplorer server containing the page *pathname* that includes the tag name.
2. Our server, specifically a JSP page, processes the request by retrieving certain information from the OpenCyc server that is somehow associated with the tag at issue.
3. The information found is parsed into tag links that are sent back as a response to TagExplorer interface that offers them to the client.

- When a tag link is clicked then the whole process repeats itself. Note that “tag link” includes our related tag links as well as del.icio.us own links between tags.

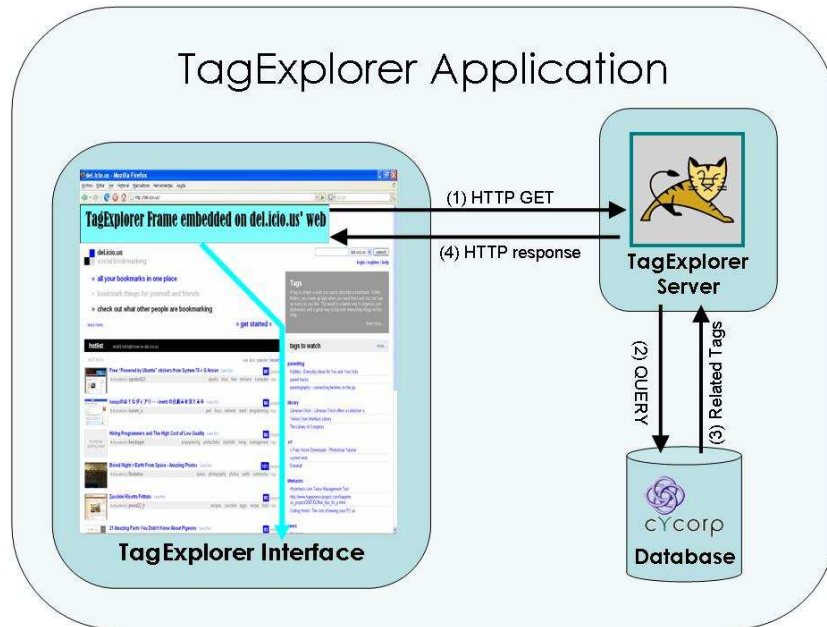


Figure 20 TagExplorer functioning

5.5 OKI Repository

The *Open Service Interface Definitions* (OSID), developed by the Open Knowledge Initiative (O.K.I), provides a variety of interfaces which comply with functions of repository, federation of repositories and digital resources. In these interfaces, there are generic calls for resolving distributed searches of resources. These OKI interfaces can be used to wrap different LUISA functionalities. In what follows, details of the main scenarios are provided.

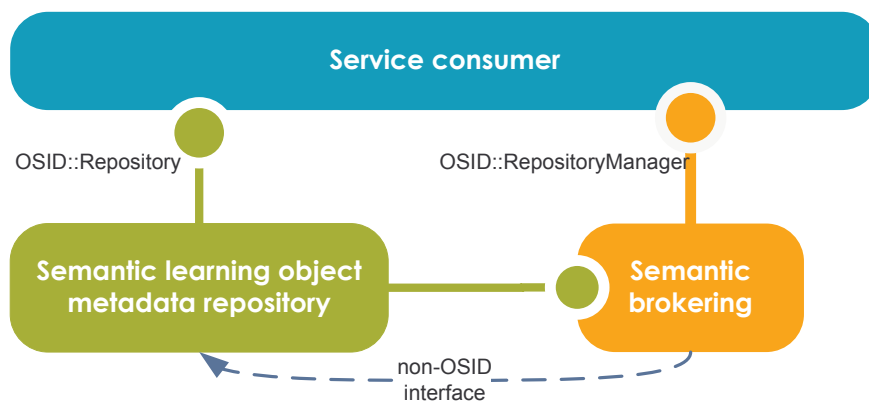


Figure 21 Approximation to support semantic search using OSID.

The OSID repository interface provides operation for different tasks including search but also creation and management of *Assets*. *Assets* are contents, metadata or combinations of both that can be given different types with different structures. The repository manager could be use to wrap LUISA distributed brokering.

Figure 21 shows that any element connected with OKI technology (*service consumer*) will access to semantic repositories using the standard interfaces *Repository* and *RepositoryManager*. Communication between semantic brokers and repositories can be done by using OSID interfaces and WSMO-based software. It is important to highlight that an OKI repository may only store metadata, so searching results can be metadata fragments that enable the location of resources, just as it is done in the LOMR.

Figure 22 shows essentials of a OKI-compliant semantic query interface. Available ontologies across the Web will be taken into account, and OKI clients can take elements from them (concepts, instances, properties...) to compose searches. In the case of LUISA, these ontologies are available through Joseki servers in the LOMR and also as part of the SWS Layer. However, in a general case, they will be available through a generic ontology repository or simply through a unique, stable URI.

Every possible combination will be defined as a *SearchType* according to OKI, which allows a great deal of flexibility. The delivery of these elements and a certain interpretation will allow the semantic repository to use different strategies of query, exploiting the knowledge represented in the ontologies.

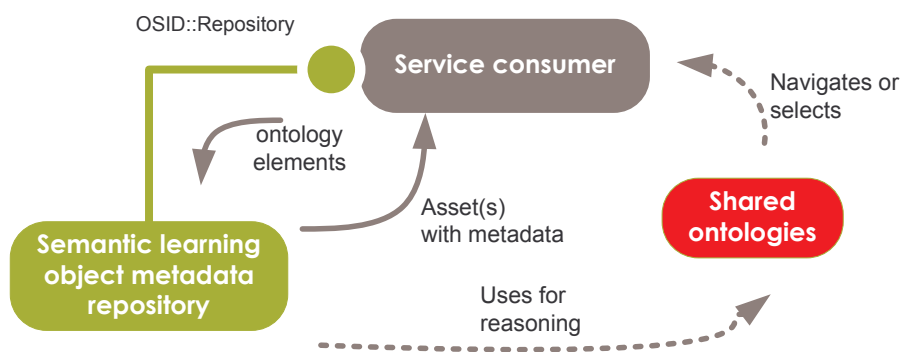


Figure 22 Semantic Query

In the case of LUISA, for example, the URIs of the competencies and other search elements could be sent through the conventional search interface of OKI, and abstracted out to a generic *SearchType* using competencies plus some additional LOM elements.

The following code fragment is an example, of the general pattern to query a repository using this interface.

```
public class ExampleOSIDRepositoryUse {
    private static LOMetadataRepository _rep;
```

```
public static void main(String[] args) throws IOException, JAXBException, Exception
{
    try {
        //Get serialized learning object
        LOM lomExample = generateExampleLOMRecord();
        LOMROKISerializedLomImpl lomExampleSerializable =
            new LOMROKISerializedLomImpl(lomExample);

        //Set repository params
        .....

        // Open the learning objects metadata repository
        _rep = LOMetadataRepositoryFactory.openLOMetadataRepository(params);
        LOMRSession sess = _rep.getLOMRAnonymousSession();

        // Set query parameters
        .....

        //Set OSID properties
        LOMROKIPropertiesImpl searchProperties =
            new LOMROKIPropertiesImpl((Serializable) parameters);

        //Adapt LOMR repository to OKI repository
        Repository okiRep = (Repository) LOMROkiAdapter.adapt(_rep);

        //get the query
        LOMROKIAssetIteratorImpl resultAssetIterator =
            (LOMROKIAssetIteratorImpl) okiRep.getAssetsBySearch
            (lomExampleSerializable, null, searchProperties);

        //get the results
        LOMROKIAssetImpl resultAsset =
            (LOMROKIAssetImpl) resultAssetIterator.nextAsset();

        //Get the learning object results
        Set<LearningObjectReference> lor =
            (Set<LearningObjectReference>) resultAsset.getContent();

    } catch (Throwable t) {
        System.out.println(t.getMessage());
    }
}
}
```

6 REFERENCES

- 1 Amorim, R. R., Lama, M., Sánchez, E., Riera, A., & Vila, X. A. (2006). "A Learning Design Ontology based on the IMS".
- 2 Reigeluth, C.M. (Ed.) (1999). "Instructional-Design Theories and Models, Volume II: A New Paradigm of Instructional Theory". Mahwah, NJ: Lawrence Erlbaum Assoc.
- 3 S.R. Chidainber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Engineering, vol. 20, no. 6, pp. 476-493, June 1994

ANNEX 1: CODE ANALYSIS RESULTS

| CLASS ANALYZED | WMC | DIT | NOC | CBO | RFC | LCOM |
|---|-----|-----|-----|-----|-----|------|
| eu.luisa.lomr.examples.lomr.BulkExportUtilityAnnotationToolEads | 2 | 1 | 0 | 6 | 18 | 1 |
| eu.luisa.lomr.examples.lomr.BulkExportUtilityAnnotationToolUhp | 2 | 1 | 0 | 6 | 18 | 1 |
| eu.luisa.lomr.examples.lomr.BulkExportUtilityStoring | 2 | 1 | 0 | 9 | 20 | 1 |
| eu.luisa.lomr.examples.lomr.BulkMetadataLoadTest | 2 | 1 | 0 | 13 | 31 | 1 |
| eu.luisa.lomr.examples.lomr.CreateNewInstanceExample | 2 | 1 | 0 | 2 | 10 | 1 |
| eu.luisa.lomr.examples.lomr.ExampleMultipleQBEMQuerying | 3 | 1 | 0 | 17 | 42 | 3 |
| eu.luisa.lomr.examples.lomr.ExampleQueryByStringEads | 2 | 1 | 0 | 9 | 21 | 1 |
| eu.luisa.lomr.examples.lomr.ExampleQueryByStringUhp | 2 | 1 | 0 | 9 | 22 | 1 |
| eu.luisa.lomr.examples.lomr.ExampleSimpleQBEMQuerying | 3 | 1 | 0 | 18 | 39 | 3 |
| eu.luisa.lomr.examples.lomr.ExampleStoring | 3 | 1 | 0 | 93 | 166 | 3 |
| eu.luisa.lomr.examples.lomr.ExampleStoringFromFolder | 5 | 1 | 0 | 6 | 46 | 10 |
| eu.luisa.lomr.examples.lomr.ExampleStoringWSMLFromFile | 2 | 1 | 0 | 7 | 15 | 1 |
| eu.luisa.lomr.examples.lomr.LOMRCreationExample | 2 | 1 | 0 | 3 | 6 | 1 |
| eu.luisa.lomr.examples.lomr.LOMRNewLOExample | 3 | 1 | 0 | 34 | 56 | 3 |
| eu.luisa.lomr.examples.nav.ExampleStoringMultipleOntologiesEads | 2 | 1 | 0 | 9 | 21 | 1 |
| eu.luisa.lomr.examples.nav.ExampleStoringMultipleOntologiesUhp | 2 | 1 | 0 | 9 | 21 | 1 |

| | | | | | | |
|---|----|---|---|----|----|-----|
| eu.luisa.lomr.examples.oki.ExampleOSIDRepositoryUse | 3 | 1 | 0 | 20 | 37 | 3 |
| eu.luisa.lomr.examples.sld.ExampleSLDQuerying | 2 | 1 | 0 | 9 | 25 | 1 |
| eu.luisa.lomr.examples.sld.ExampleSLDQueryingTheories | 2 | 1 | 0 | 9 | 24 | 1 |
| eu.luisa.lomr.examples.sld.ExampleStoringSLDOntology | 2 | 1 | 0 | 4 | 6 | 1 |
| eu.luisa.lomr.LearningObjectReference | 2 | 1 | 0 | 0 | 2 | 1 |
| eu.luisa.lomr.LOMetadataRepository | 5 | 1 | 0 | 2 | 5 | 10 |
| eu.luisa.lomr.LOMRQBEManager | 1 | 1 | 0 | 3 | 1 | 0 |
| eu.luisa.lomr.LOMRQueryManager | 33 | 1 | 0 | 2 | 33 | 528 |
| eu.luisa.lomr.LOMRSession | 2 | 1 | 0 | 2 | 2 | 1 |
| eu.luisa.lomr.nav.LOMRNavigationalQueryManager | 1 | 1 | 0 | 1 | 1 | 0 |
| eu.luisa.lomr.nav.LOMRNavigationalSessionManager | 13 | 1 | 0 | 5 | 13 | 78 |
| eu.luisa.lomr.nav.wsml.LOMRNavigationalOntologyQuery | 26 | 1 | 0 | 10 | 65 | 247 |
| eu.luisa.lomr.nav.wsml.LOMRNavigationalOntologyQuery | 1 | 3 | 0 | 0 | 2 | 0 |
| eu.luisa.lomr.nav.wsml.LOMRNavigationalQueryManagerWsml | 3 | 1 | 0 | 16 | 17 | 1 |
| eu.luisa.lomr.nav.wsml.LOMRNavigationalSessionManagerWsml | 15 | 1 | 0 | 14 | 54 | 0 |
| eu.luisa.lomr.nav.wsml.util.ConsoleNavigationalInterface | 2 | 1 | 0 | 14 | 48 | 1 |
| eu.luisa.lomr.nav.wsml.util.LOMRNavigationalConfiguration | 8 | 1 | 0 | 4 | 23 | 0 |
| eu.luisa.lomr.nav.wsml.util.LOMRNavigationalReasoner | 4 | 1 | 0 | 20 | 39 | 0 |
| eu.luisa.lomr.nav.wsml.util.RepositoryCreation | 2 | 1 | 0 | 8 | 16 | 1 |
| eu.luisa.lomr.oki.LOMROkiAdapter | 31 | 1 | 0 | 26 | 51 | 465 |

| | | | | | | |
|--|-----|---|---|----|-----|------|
| eu.luisa.lomr.oki.LOMROKIID | 3 | 1 | 0 | 2 | 7 | 0 |
| eu.luisa.lomr.oki.LOMROKIRepositoryImpl | 30 | 1 | 0 | 26 | 60 | 423 |
| eu.luisa.lomr.oki.LOMROriginalLom | 2 | 1 | 0 | 84 | 295 | 0 |
| eu.luisa.lomr.oki.wsml.LOMROKIAssetImpl | 32 | 1 | 0 | 13 | 35 | 494 |
| eu.luisa.lomr.oki.wsml.LOMROKIAssetIteratorImpl | 3 | 1 | 0 | 5 | 6 | 1 |
| eu.luisa.lomr.oki.wsml.LOMROKIPropertiesImpl | 4 | 1 | 0 | 4 | 6 | 4 |
| eu.luisa.lomr.oki.wsml.LOMROKISerializedLomImpl | 137 | 1 | 0 | 82 | 231 | 8718 |
| eu.luisa.lomr.ordi.wrappers.JenaImpl | 15 | 1 | 0 | 17 | 39 | 0 |
| eu.luisa.lomr.ordi.wrappers.JenaTripleImpl | 7 | 1 | 0 | 21 | 42 | 13 |
| eu.luisa.lomr.ordi.wrappers.JenaTripleIteratorImpl | 4 | 1 | 0 | 5 | 9 | 0 |
| eu.luisa.lomr.QBEQueryResult | 0 | 1 | 0 | 1 | 0 | 0 |
| eu.luisa.lomr.QueryExplanation | 2 | 1 | 0 | 1 | 2 | 1 |
| eu.luisa.lomr.QueryResult | 2 | 1 | 0 | 0 | 2 | 1 |
| eu.luisa.lomr.sld.SLDQueryManager | 9 | 1 | 0 | 2 | 9 | 36 |
| eu.luisa.lomr.sld.wsml.SLDQueryManagerImpl | 15 | 0 | 0 | 19 | 61 | 15 |
| eu.luisa.lomr.sld.wsml.SLDQueryResultImpl | 4 | 1 | 0 | 1 | 16 | 0 |
| eu.luisa.lomr.sld.wsml.util.LOMRLearningDesignReasoner | 3 | 1 | 0 | 20 | 41 | 1 |
| eu.luisa.lomr.translator.lom2wsml.adapter.AriadneAdapter | 3 | 1 | 0 | 0 | 20 | 3 |
| eu.luisa.lomr.translator.lom2wsml.adapter.CareoAdapter | 2 | 1 | 0 | 0 | 18 | 1 |
| eu.luisa.lomr.translator.lom2wsml.ElementTranslator | 1 | 1 | 0 | 5 | 1 | 0 |

| | | | | | | |
|---|---|---|----|----|----|---|
| eu.luisa.lomr.translator.lom2wsml.LOM2WSMLTranslator | 0 | 1 | 0 | 1 | 0 | 0 |
| eu.luisa.lomr.translator.lom2wsml.Metadata2OntologyTranslator | 1 | 1 | 0 | 0 | 1 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.A2ATranslator | 2 | 0 | 0 | 26 | 38 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.C2CTranslator | 2 | 0 | 0 | 28 | 45 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.C2MITranslator | 2 | 0 | 0 | 18 | 24 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.CL2CLTranslator | 2 | 0 | 0 | 26 | 39 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.CO2COTranslator | 2 | 0 | 0 | 16 | 24 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.D2DTranslator | 2 | 0 | 0 | 22 | 34 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.DirectTranslator | 2 | 0 | 0 | 15 | 24 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.ElementTranslatorImpl | 5 | 1 | 16 | 10 | 19 | 4 |
| eu.luisa.lomr.translator.lom2wsml.wsml.I2LITranslator | 2 | 0 | 0 | 18 | 28 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.L2HLTranslator | 2 | 0 | 0 | 17 | 27 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.LOM2WSMLTranslatorImpl | 4 | 1 | 0 | 30 | 45 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.LS2LSTranslator | 2 | 0 | 0 | 27 | 40 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.MD2MDTranslator | 2 | 0 | 0 | 34 | 50 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.MT2MTTTranslator | 2 | 0 | 0 | 17 | 25 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.R2RTranslator | 2 | 0 | 0 | 22 | 34 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.TAG2TAGTranslator | 2 | 0 | 0 | 18 | 27 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.TR2TRTranslator | 2 | 0 | 0 | 23 | 33 | 0 |
| eu.luisa.lomr.translator.lom2wsml.wsml.VI2VITranslator | 2 | 0 | 0 | 30 | 39 | 0 |

| | | | | | | |
|--|----|---|---|----|----|----|
| eu.luisa.lomr.wsml.LearningObjectReferenceFactory | 2 | 1 | 0 | 4 | 7 | 1 |
| eu.luisa.lomr.wsml.LearningObjectReferenceImpl | 6 | 1 | 0 | 1 | 11 | 0 |
| eu.luisa.lomr.wsml.LOMetadataRepositoryConfigException | 1 | 3 | 0 | 0 | 2 | 0 |
| eu.luisa.lomr.wsml.LOMetadataRepositoryFactory | 3 | 1 | 0 | 3 | 5 | 3 |
| eu.luisa.lomr.wsml.LOMetadataRepositoryImpl | 10 | 1 | 0 | 25 | 43 | 23 |
| eu.luisa.lomr.wsml.LOMRQBQueryManagerImpl | 14 | 0 | 0 | 49 | 99 | 0 |
| eu.luisa.lomr.wsml.LOMRQueryManagerImpl | 34 | 1 | 2 | 17 | 88 | 0 |
| eu.luisa.lomr.wsml.LOMRSessionImpl | 3 | 1 | 0 | 8 | 13 | 0 |
| eu.luisa.lomr.wsml.QBQueryResultImpl | 3 | 1 | 0 | 2 | 8 | 0 |
| eu.luisa.lomr.wsml.util.BulkExportUtility | 2 | 1 | 0 | 10 | 27 | 1 |
| eu.luisa.lomr.wsml.util.EadsWSEExample | 2 | 1 | 0 | 7 | 58 | 1 |
| eu.luisa.lomr.wsml.util.LomDuration | 3 | 1 | 0 | 0 | 10 | 3 |
| eu.luisa.lomr.wsml.util.UhpWsExample | 2 | 1 | 0 | 7 | 59 | 1 |
| eu.luisa.lomr.wsml.util.WSLom | 2 | 1 | 0 | 7 | 58 | 1 |
| eu.luisa.lomr.wsml.util.WSMLJenaConsoleOutput | 3 | 1 | 0 | 8 | 26 | 3 |