Model Driven E-learning Platform Integration

Zuzana Bizonova, Daniel Ranc, Matilda Drozdova
Expected graduation: September 2008
Advisors' names: Daniel Ranc (INT Evry, France),
Matilda Drozdova (Zilina University, Slovakia)

Abstract. The success of the e-learning paradigm observed in recent times created a growing demand for e-learning systems in universities and other educational institutions, that itself led to the development of a number of either commercial or open source learning management systems (LMS). While the usage of these systems gains recognition and acceptance amongst institutions, there are new problems arising that need to be solved. Because of multiplicity of platforms and approaches used for various systems implementation, it becomes increasingly difficult to manage or compare them. Their variety and growing number is also a true barrier for re-use of existing learning materials that is a clear economical concern for the future of these technologies. Applications and their data become isolated. As the result of platform diversity, future vision of interconnecting LMS of different educational institutions is demanding, too.
The present study ambitions to overcome the aforementioned difficulties by using the Model Driven Architecture (MDA) approach of the Object Management Group (OMG). The goal is to provide a generalized architectural framework enabling an integrated specification of platform architectures. This platform-independent framework can then be used to specify and classify existing or future Learning management systems and to simplify migration of data between different kinds of e-learning systems.

Keywords: Model Driven Architecture, Learning Management Systems, Integration.

1 Introduction

As a result of Technology Enhanced Learning (TEL) boom during the last few years, many educational institutions all over the world started to use Learning Management Systems (LMS) to manage e-education of their students. TEL became very popular because of many advantages it offers comparing to the classical way of education – it is possible to study whenever and wherever the student wants to, with the individual speed and use a lot of multimedia interactive material for the studies.

As the answer of sudden popularity of e-learning, number of Learning Management Systems were developed. LMS should make it easy to publish documents, lectures and exercises for professors, assistants and secretaries at the educational institute and faculty. Besides, it should be the main portal for all the students to get the recent information for their courses and exercises. Many institutions created their own LMS, some of them bought a commercial system or adopted an open-source solution.

Architecture of a system consists of elements and relationships between them. Although at the end of the day functionalities of the systems resemble, their internal structure is often completely different. As a result of that applications and their data become isolated. It is also demanding to compare the systems because they use different kind of terminology for describing the same functionality.

The fact that LMS systems have different architectures causes multiple practical problems. It is difficult to share information among systems. This is crucial for example in learning material re-use or students' records sharing among universities. Currently it is difficult to use learning object from an LMS in another one and it can cause great economical problems in the future of these technologies. We should think hard how to enable re-use of already existing material so that we don't need to re-create the same materials again. The students' records sharing among universities can be used for example in a centralised data system among universities where student can choose courses from various institutions. Furthermore to shuffle from one system to another is difficult, because information from one system cannot be easily transferred to another one. Moreover, it is complicated to compare LMS systems because they use different kind of terminology for describing the same functionality.

This project suggests a solution of this situation by using the Model Driven Architecture (MDA) [MDA] approach of the Object Management Group (OMG). We would like to create a common architectural framework enabling an integrated specification of platform architectures. This platform-independent framework can then be used to specify and classify existing or future Learning management systems and to simplify the courseware material re-use from different kinds of e-learning systems.

In the following section we introduce MDA, its fundamental model concepts and relationships between these concepts and afterwards a possible solution of the proposed problem: a staged approach to the Platform Independent Model of MDA. In the third section there is an overview of principles of two open source e-learning systems, Moodle and OLAT. They serve as examples of learning management systems with different architectures and technologies but similar functionalities. The fourth section shows the integration strategy in the example of Moodle and OLAT. Finally, the last section describes concluding remarks and future work.

2 Model Driven Architecture and Its Use For LMS Integration

MDA is a way to organize and manage system architectures; it is supported by automated tools and services for both defining the models and facilitating model types [1].

The MDA approach was proposed by OMG, the open standard organization supporting the well-known CORBA [5] and Unified Modeling Language (UML) [2]. The models in MDA [3] may be developed as a precursor to implement the physical system, or they may be derived from an existing system or a system in development as an aid to understand its behaviour. The building of the system can be organized around a set of models by imposing a series of transformations between them. The whole system creates an architectural framework of layers and transformations.

OMG defines three types of models [4] :

- Computation Independent Model (CIM) – this model is focused on the domain, hiding structure details,

- Platform Independent Model (PIM) – this model provides adequate functionalities, structure and behavior of the system,
- Platform Specific Model (PSM) – combines PIM with specific detail concerning the way in which the system uses a certain platform – it can be automatically transformed into the implementation code.

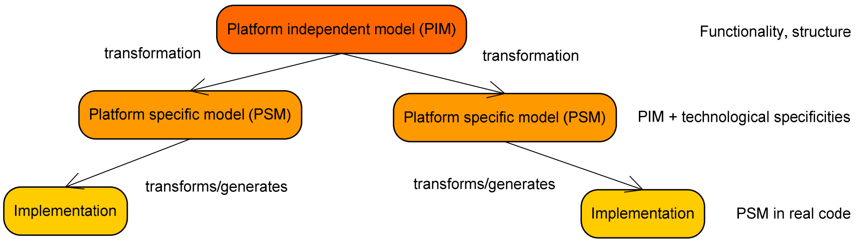

Fig. 1. MDA Concept

Usage of MDA to build up system architectures has several advantages. It consists of models at varying levels of abstraction, which means that refinements of the models are possible at any level. This approach helps users to get a very clear idea of the system. Models help people understand and communicate complex ideas. This way, we can see the commonalities and differences of systems at all levels.

We would like to use MDA principle as the background for solution of the proposed problem with LMS integration. We can compare platform independent models of different systems and create a platform independent model that covers common functionalities of all learning management systems.

Our goal is to define a generalized model of LMS system consisting of features of all other LMS systems that can be mapped into it. For this purpose we introduced a new strategy [10] that we call the *reversed MDA paradigm*. In classical MDA approach we create first an abstract PIM model and with automatic steps we can get a PSM and finally the implementation code. In our case we would like to go the other way round – to use real LMS systems to define and abstract model of a generalised LMS.

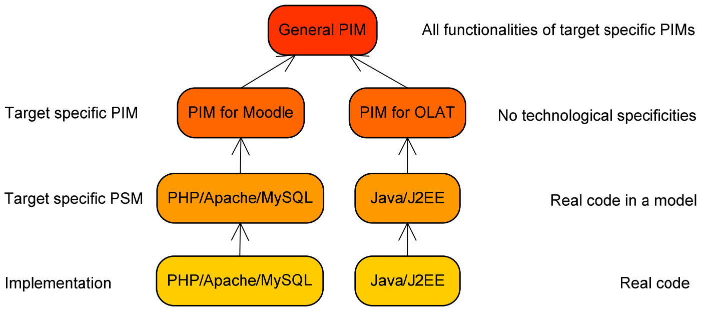

Fig. 2. Reverse MDA Concept for creation of General PIM

The creation of the General PIM needs to be done in several steps. We come out from the premise that functionality is similar for most of the LMS systems thought their implementation and internal architecture is different. We continually add functionalities of different LMS systems until the General model is saturated.

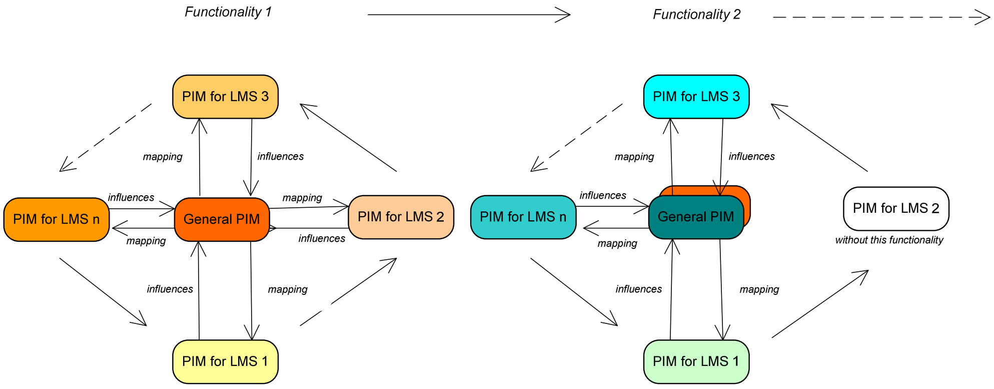

Fig. 3. Integration strategy

The concrete example of the integration strategy will be presented it the chapter four. It will be a real-life example and for this purposes we have chosen two different open source LMS systems described in the following paragraphs.

3 Overview of Two Opensource LMS

To compare the differences between two approaches of LMS systems, two open source systems have been chosen: Moodle based on PHP and OLAT that represents a Java solution.

OLAT [6] is a web-based open source LMS that was founded in 1999 at the University of Zurich, Switzerland. OLAT is implemented in Java and uses a three-tier architecture with Tomcat container technology. Regarding programming concepts, OLAT is a component based tool. Component visually represents for instance a form or a table. OLAT was designed to separate the logic of the application and the layout of the web site. It proposes a refined Model-View-Controller scheme where usage logic is encapsulated in controllers and the manager classes they use, while layout is controlled by modifying Cascading Style Sheets (CSS).

Moodle [8] is an open source software package that was founded in the same year as OLAT, in 1999, in East Perth, Australia. Moodle is implemented in PHP, uses a traditional Apache server and a relational database management system. Therefore the layout of the web site is not separated from the logic of the system. PHP is not an object oriented language in comparison to Java, therefore Moodle is implemented without objects.

The two LMSs represent very different architectural breeds that make them good candidates for our purposes.

4 Integration Strategy in Practise

In this section we would like to present an example of the integration strategy that is used in our project. LMS1 in our example is OLAT, as LMS2 we use Moodle.

In the previously published articles we have had a closer look at a certain set of functionalities - access rights [11] to the LMS systems and learning object management [12]. This time we will focus on explaining the integration strategy at those two sets of functionalities with OLAT and Moodle.

Functionality 1: Access rights

In OLAT [7] we can recognize five kinds of users while there are four different ones in Moodle. It is possible to recognize the rights of the users on the Figure 4:

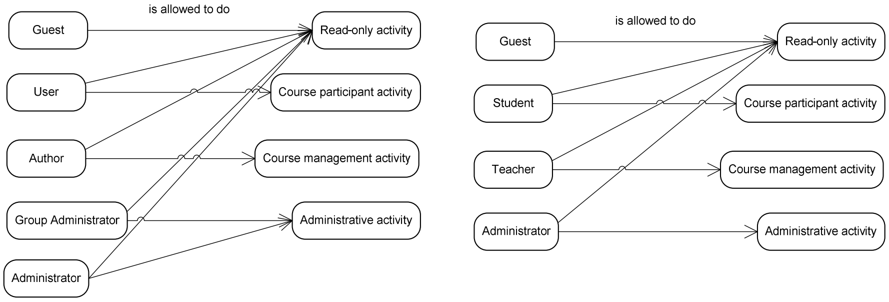

Fig. 4. Access rights for different kinds of users in OLAT and Moodle.

However, the roles are not the only attribute that determines access rights in Moodle. To enable greater flexibility, for example to allow a student with role A to post to forums, while student with role B is prevented from posting, three more attributes are defined: capability, permission and context. The role consists of a list of capabilities for different possible actions within Moodle. For example a teacher is allowed to add learning resources to a course A but is prevented from adding resources to another course where he is considered to be a student.

We can show the integration into a General model in the proposed analysis. The General model integrates approaches from both systems. All users in both systems are allowed to do read-only activities, as can be seen on Figures 4 and 5. Therefore the parent class G_Role contains read-only procedures that are inherited by all children classes: G_Guest, G_Student, G_Teacher and G_Admin. The children classes contain procedures that are allowed for a guest, a student, a teacher or an admin (Figure 5).

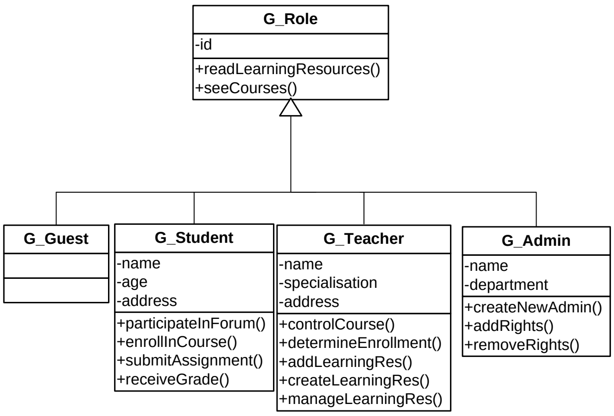

Fig. 5. Classes of General Platform Independent Model. This model shows allowed activities for each class.

The classes of Moodle and OLAT can be mapped into the classes of General PIM. Please see the details in the table 1. The previously mentioned roles A and B of student in Moodle can be modeled as child classes of Student class, Student::StudentA and Student::StudentB

Table 1. Mapping the classes from Moodle and OLAT to the General PIM.

General PIM	Moodle	OLAT
G_Guest	Guest	Guest
G_Student	Student	User
G_Teacher	Teacher	Author
G_Admin	Administrator	Group Administrator, Administrator

Functionality 2: Learning object management

After the General PIM model is saturated – all observed LMSs are mapped into it and it is mapped to all the observed LMSs – it is possible to continue with another set of functions.

As another example, relevant for learning object re-use, we consider part of the learning management systems that incorporates learning objects. In a regular system, users with different access rights to learning objects can view them, add them, edit them, catalogue them and in some cases, import them from other systems, export them and search for them. The searching possibility is not a regular part of an LMS system and only a few of them have this possibility by default, for example OLAT repository.

Slightly simplified PIM models of both LMS systems are modeled on the Figures 6 and 7, OLAT and Moodle respectively. Here we can see the objects of both systems and relationships between them.

Each resource of the repository in OLAT is an instance of the class *RepositoryEntry* that contains attributes like the name of the resource, its location, the author or activities that are allowed for the resource. The list of attributes can be broaden in the *MetadataElement* class in which we can define any other metadata, with their name and value (for example name = version, value = 1.2). Repository entries can be ordered in a catalogue, with the help of *CatalogEntry* class. All the entries have an id defined in the OLAT system in the *OLATResorceImpl* class and each resource can point to other resources via *ReferenceImpl* class. The permissions of a user to do different kind of activities with the resource are noted in *PolicyImpl* class.

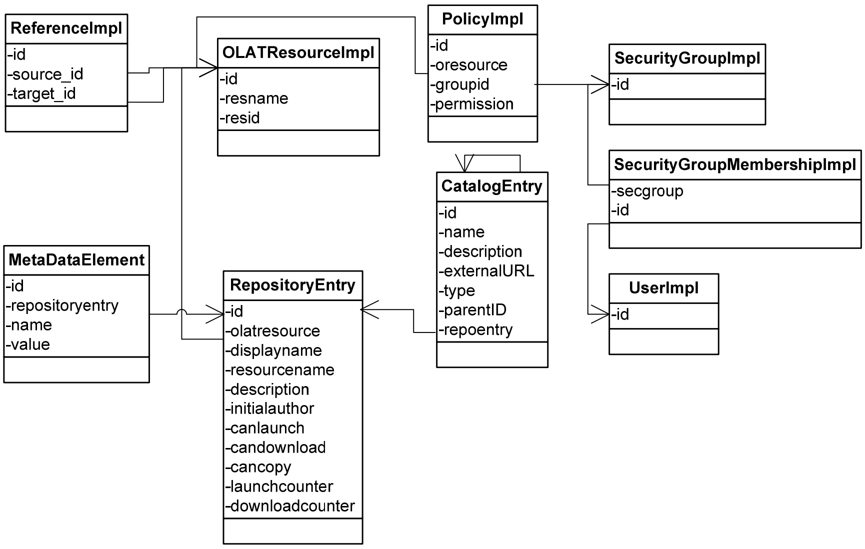

Fig. 6. LMS specific PIM for OLAT

In Moodle, in comparison, there is a course oriented system - each Course contains a list of resources of different types. They can be ordered with the help of *CourseMetaData* and they can be displayed to a User according to a *CourseDisplay* table.

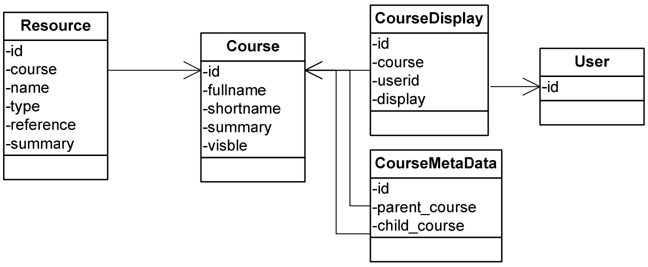

Fig. 7. LMS specific PIM for Moodle

Based on the PIM models of the systems we created a General PIM (see Figure 8) that contains the descriptions of both systems. The attributes of the resources (for example *General_Title*, *Technical_Size*) are based on IEEE Learning Object Metadata standard (LOM) [9]. This standard defines a set of elements ordered in categories. They can be used to describe learning resources. IEEE LOM is a part of SCORM specification and became standard of IEEE Computer ociety/Learning Technology Standards Committee in 2002.

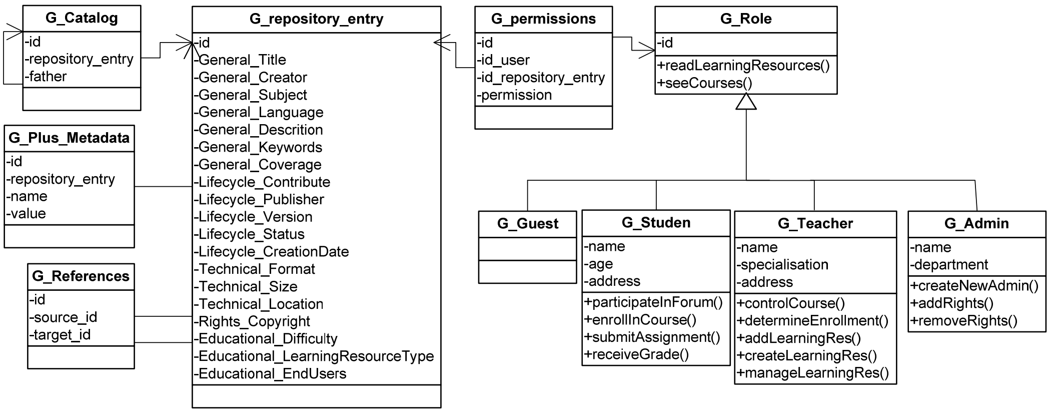

Fig. 8. General PIM – metadata are based on IEEE Learning Object Metadata

The repository entries can be ordered in a catalog (G_Catalog), they can be added any amount of extra metadata (G_Plus_Metadata), they can point the reference to any other entry (G_Reference) and they and they can be viewed and copied based on permissions of a given user (G_permissions). It is possible to map the attributes and relationships between models of LMS systems and General PIM based on certain mapping rules [12].

In this section we have proposed an example of how the integration strategy can be used for creation of the generalized PIM. The example has been shown on two opensource LMS systems – OLAT and Moodle in two sets of functions – one considering access rights, the other one considering management of learning objects in each of these systems. Figure 8 shows the model after these two steps. We have defined not only the model but also mapping of architectures of OLAT and Moodle from and to the General model.

5 Conclusion

This contribution presents an original approach to the problem of integrating LMSs of different architectures. In particular it proposes an integration strategy of LMSs into a generalized model of a LMS with an example of this integration strategy at two sets of functionalities – access rights and learning object management. In the example we modelled a PIM of two LMS systems (Moodle and OLAT) and showed how to map them to a generic General PIM. The framework still needs to be enriched by more examined LMS systems.

This staged concept is viewed as a foundation for providing ultimately an integrating LMS MDA model, with the goal to solve current challenges related to the multiplicity of the platforms, such as LMS management, comparison, and data share.

References

1. Brown, A.: An introduction to Model Driven Archtiecture, IBM developer works (Feb 2004), http://www-128.ibm.com/developerworks/rational/library/3100.html
2. Dr. Jon Siegel, OMG: Introduction to OMG's Unified Modeling Language (July 2005), http://www.omg.org/gettingstarted/what_is_uml.htm
3. Eleed, University of Muenster, H. L. Grob, F. Bensberg, B. L. Dewanto: Model Driven Architecture (MDA): Integration and Model Reuse for Open Source eLearning Platforms, (2004) http://eleed.campussource.de/archive/1/81/
4. N. Moreno, J.R.Romero: A MDA-based framework for building interoperable e-learning platforms, Recent Research Developments in Learning Technologies, Universidad de Málaga, Spain (2005), http://www.formatex.org/micte2005/193.pdf
5. John Wiley: CORBA – Fundamentals and Programming, John Wiley and Sons Publishing (1996).
6. OLAT Project, University of Zurich, Switzerland, http://www.olat.org, 1999.
7. Florian Gnägi: Olat 4.0 – Overview of functions, University of Zurich (Nov 2005), http://www.olat.org/downloads/material/OLAT_4_0_Overview_of_functions_v15.pdf
8. Moodle project, Moodle Developper Documentation (Nov 2006), http://docs.moodle.org/en/Developer_documentation
9. Learning Object Metadata Standard, IEEE (2002), http://ltsc.ieee.org/wg12/
10. Bizonova, Z., Ranc, D.: Integrácia Learning Management Systémov pomocou MDA, Conference Technologie pro e-vzdělávaní 2007, Praha, (June 2007)
11. Bizonova, Z., Ranc, D.: Model Drivem LMS Platform Integration, The Third advanced International Conference on Telecommunications, (May 2007) http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/aict/2007/2843/00/2843toc.xml&DOI=10.1109/AICT.2007.29
12. Bizonova, Z., Ranc, D.: Courseware Material Re-use via Model Driven LMS Platform Integration, CBLIS conference 2007, (July 2007)