COURSEWARE MATERIAL RE-USE VIA MODEL DRIVEN LMS PLATFORM INTEGRATION

Zuzana Bizoňová, Daniel Ranc

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 e-learning platforms, as well as of relevant standards (e.g. SCORM). While the usage of these platforms gains recognition and acceptance amongst institutions, two difficulties arise that are directly related to the diversity of available platforms. On one hand, it becomes increasingly difficult to manage or to compare available platforms without additional conceptual tools. On the other hand the growing multiplicity of different platforms is a true barrier to the re-use of existing course material, which is a clear economical concern for the future of these technologies. The present project 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 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 (LMS) and to simplify the courseware material re-use from different kinds of e-learning systems.

KEYWORDS
E-learning platform integration, Learning management systems, Model driven architecture, Courseware material re-use

INTRODUCTION

During the last few years with the great growth of Internet many educational institutions all over the world have started to use various learning management systems (LMS). These systems became very popular because of many advantages they offer comparing to the classical way of education. To mention just a few of them, with LMS it is possible to:

- study from multimedia resources (animations, sounds or movies help a student to understand the topic better and deeper),
- study with the speed that is suitable for each student,
- study whenever and wherever,
- test your abilities during and after finishing the course,
- communicate with the teacher and other students not only in the classroom but also on Internet forums and by email.

Certainly, on one hand, e-learning has offered many new possibilities to the educational system. On the other hand, to create multimedia resources for the course can mean a lot of extra effort for the teachers. Of course it is very helpful for a student to see an animation that describes the signal in telecommunication networks, but to create such an animation may take many hours. Therefore, when somebody creates such a resource, many other teachers would prefer to use it during their lessons.
A learning resource becomes more valuable if it can be used for many other courses as well. However, the growing multiplicity of different platforms is a true barrier to the re-use of existing course material

(Grob, Bensberg, Dewanto, 2004). Often, educational institutions use various LMS systems that do not cooperate although their functionalities are only slightly different.

In this project we would like to introduce a unique solution to this problem by using the Model Driven Architecture (MDA) approach of the Object Management Group (OMG, 2003). We will define a formal integration framework for LMS systems, as a basic architecture for sharing material among various platforms.

In the following section we introduce MDA, its fundamental model concepts and relationships between these concepts. In the third section there is an overview of principles of two open source e-learning systems, Moodle (Moodle, 2006) and OLAT (OLAT, 1999). They serve as examples of learning management systems with different technologies and architectures but similar functionalities. The fourth section proposes a possible solution to the proposed problem, introducing a staged approach to the Platform Independent Model of MDA. Finally, the last section describes concluding remarks and future work.

MODEL DRIVEN ARCHITECTURE

Model Driven Architecture (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 (Brown, 2004).

The MDA approach was proposed by OMG, the open standard organization supporting the well-known CORBA (Wiley, 1996) and Unified Modelling Language (UML) (Siegel, 2005). The models in MDA (Grob, Bensberg, Dewanto, 2004) 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.

MDA uses the separation of concerns. It is intended to separate usage-oriented decisions from platform decisions to allow greater flexibility throughout the lifecycle of these systems.
It is convenient to allow users to express system perspectives of value to software architects and developers in ways that are readily mapped into programming language. The MDA approach involves creating abstract models and describing them with standard specification languages so that an implementation can be generated almost automatically.

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 (Moreno, Romero, 2005)

- Computation Independent Model (CIM) – this model is focused on the domain, hiding structural details,
- Platform Independent Model (PIM) – this model provides adequate functionalities, structure and behaviour of the system, and
- Platform Specific Model (PSM) – combines PIM with specific detail concerning the way in which the system uses a certain platform.

The “platform” in this case is a relative term that depends on the context. For example, a single model can be a CORBA-specific PSM, because we decide to use CORBA (Wiley, 1996) as middleware, but it is also a PIM with respect to operating system and hardware.

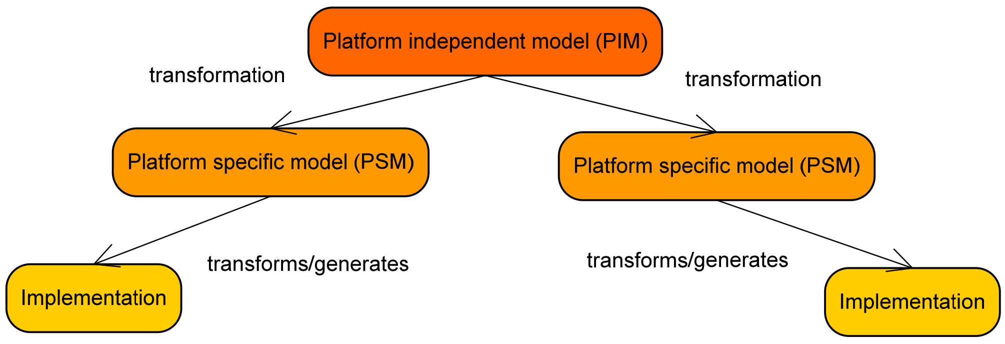

Figure 1. MDA Concept

Using MDA to build up system architectures has several advantages. Firstly, it consists of models at varying levels of abstraction, this 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 can help people to understand and communicate complex ideas. They can see the commonalities and differences of systems at all levels. Secondly, it is possible to transform automatically the models to actual implementation code. The models may serve as the basis for software architectures and the transformations may be largely automated.

Last but not least, MDA relies on open standards protecting projects against technology changes. Some Open standards that MDA uses for modelling (OMG, 2003) are:

- the Unified Modelling Language (UML) – It is the suggested modelling notation for PIM/PSM but it is not compulsory to use it. Currently there is increasing interest in development of Domain Specific Language (DSL (Compose Project, 2004)).
- the Meta-Object Facility (MOF) (Siegel, 2005) – standard language for expressing metamodels. A metamodel uses MOF to formally define the abstract syntax of a set of modelling constructs.
- XML Metadata Interchange (XMI) (OMG, 2003) - is a model driven XML Integration framework for defining, interchanging, manipulating and integrating XML data and objects. XMI provides a mapping from MOF to XML.
- Transformation is possible into most platforms like CORBA, J2EE, .NET and web based platforms.

To conclude, the MDA approach provides a conceptual framework and a set of standards to express models, model relationships, and model-to-model transformations that serve as the basis for software architectures that are ultimately realized through various implementation technologies. We can use this principle as the background for solution of the proposed problem with LMS integration. We can compare platform specific models of systems and create a platform independent model that covers common functionalities of all learning management systems. As examples of LMS we use two open source systems: Moodle and OLAT.

TWO EXAMPLES OF OPEN SOURCE LMS SYSTEMS

Although most LMS have similar functionalities, platform integration and comparison of the systems can be difficult because of the different web-based technologies used. Most of the open source systems are based on PHP, while just a minority are implemented using Java or J2EE. To compare the differences between these two approaches we have chosen two open source learning management systems: Moodle based on PHP and OLAT that represents a Java solution.

OLAT (OLAT 1999) 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. A 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 (Moodle, 2006) 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, and uses a traditional Apache server with 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.

COMMON LMS FRAMEWORK DESIGN

Our project requires defining an integration strategy. In traditional MDA, we identify a platform independent model, further transformed into a platform-specific model that can be implemented. This generic approach works perfectly in an environment where the system has homogeneous functionalities. However, in the proposed study, the functional diversity of the analyzed systems has to be taken into consideration.

Therefore, we propose to decompose the first PIM step of MDA, into two model substeps: a *common PIM model*, and a *target-specific PIM model*.

The common PIM models all existing common functionalities of the studied LMSs.

The common PIM can be further mapped into target-specific PIMs that are tailored to each particular LMS. The target-specific PIMs contain such functionalities and classes that cannot be mentioned in the common metamodel and the classes themselves can be extended here. All target- specific PIMs are transformed to platform specific models with different technologies employed in each LMS and finally the implementation could be generated.

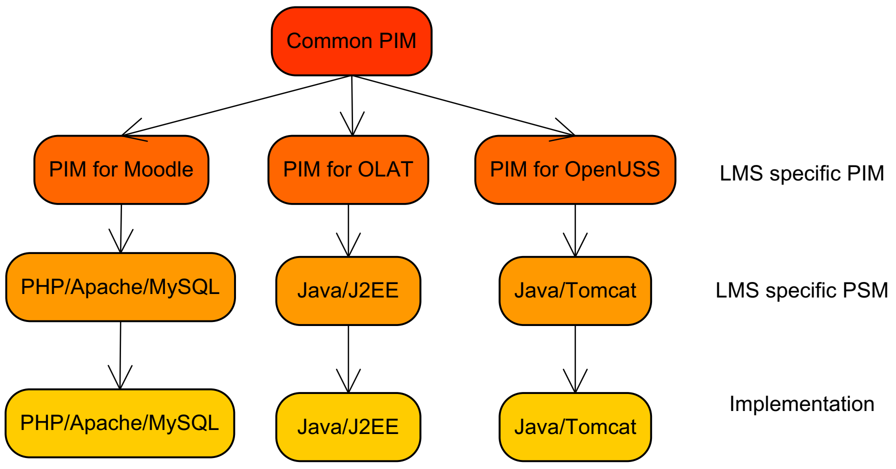

Figure 2. MDA Application on LMS Platform Integration

It has to be noted that this staged approach focuses on functionality, not on technology, and as such does not breach the platform independence contract of the PIM.

The benefit of this approach is to integrate various learning management systems into one common metamodel that can be translated into any platform. Moreover, the LMS-specific PIMs allow a detailed visibility of functional differences or specificities of the underlying LMSs. Therefore it could be easier to compare the functionalities between the systems simply by comparing PIMs of different LMSs.

As an 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, the OLAT repository.

Slightly simplified PIM models of both LMS systems are modelled on the Figures 3 and 4, OLAT and Moodle respectively. Here we can see the objects of both systems and relationships between them. Although Moodle is implemented in PHP that is not by default an object oriented language, objects are derived from the entity-relation model of the system and there are ways to integrate objects in a PHP application as well.

Each resource of the repository in OLAT is an instance of class the *RepositoryEntry* that contains attributes like name of the resource, its location, 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 the *CatalogEntry* class. All the entries have an id defined in the OLAT system in the *OLATResourceImpl* class and each resource can point to other resources via the *ReferenceImpl* class. The permissions of a user to do different kind of activities with the resource are noted in the *PolicyImpl class*.

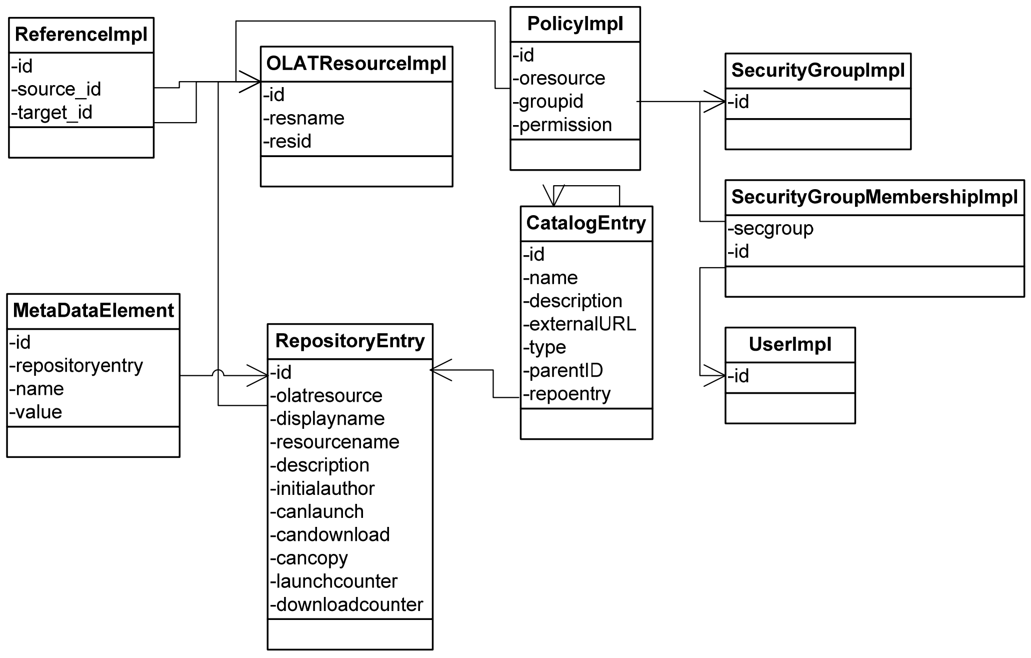

Figure 3. LMS specific PIM for OLAT

In the Moodle 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.

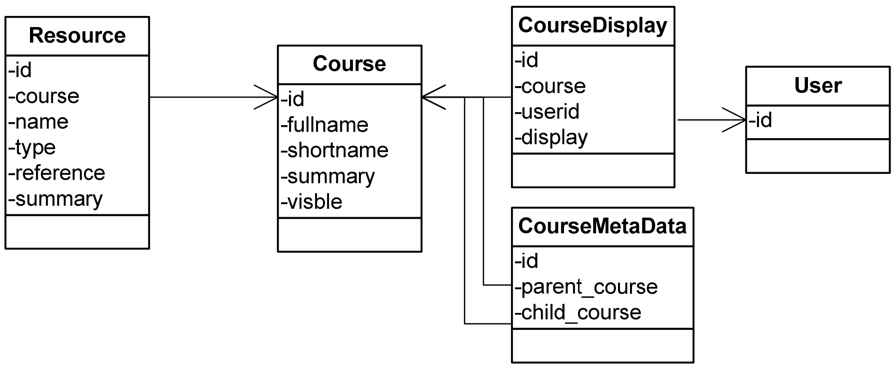

Figure 4. LMS specific PIM for Moodle

Based on the PIM models of the systems we created a common PIM (see Figure 5) that contains the functionalities of both systems. The attributes of the resources (for example *General_Title*, *Technical_Size*) are based on IEEE Learning Object Metadata standard (LOM) (IEEE, 2002). 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 (SCORM, 2004) specification and became standard of the IEEE Computer Society/Learning Technology Standards Committee in 2002.

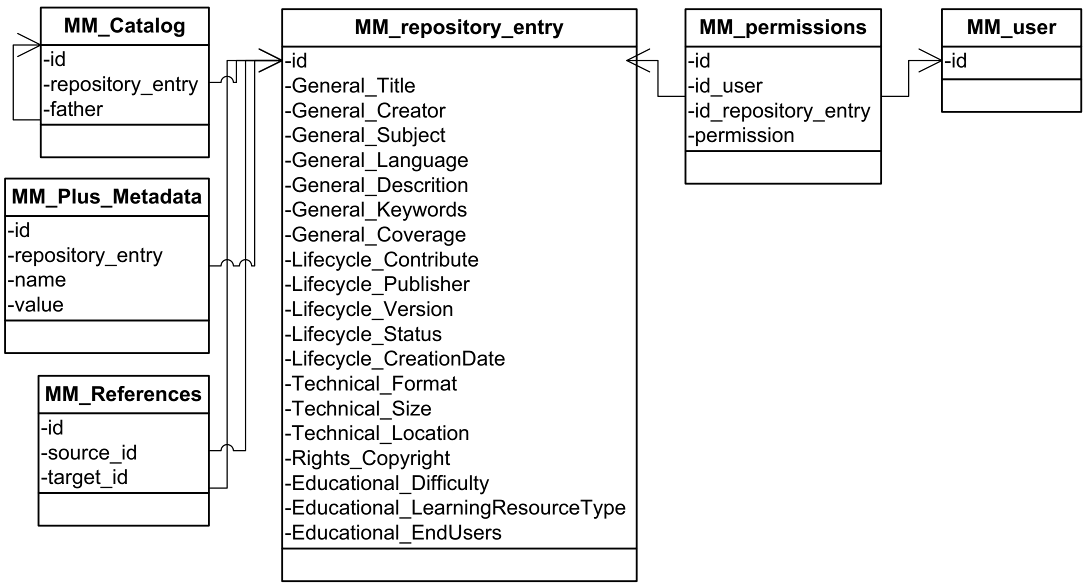

Figure 5. Common PIM – MetaModel – metadata are based on IEEE Learning Object Metadata

The repository entries can be ordered in a catalog (*MM_Catalog*), and any amount of extra metadata can be added (*MM_Plus_Metadata*), they can point the reference to any other entry (*MM_Reference*) and they and they can be viewed and copied based on permissions of a given user (*MM_permissions*).

From the common PIM to other platform specific PIMs we can map the attributes and relationships based on the rules described in tables 1, 2, 3 and 4.

Metamodel	**OLAT**	**Moodle**
General_Title	RepositoryEntry.displayname	Course.fullname / Resource.name
General_Creator	RepositoryEntry.initialauthor	(blank)
General_Subject	(MetadataElement)	(blank)
General_Language	(MetadataElement)	(blank)
General_Description	RepositoryEntry.description	Course.summary / Resource.summary
General_Keywords	(MetadataElement)	(blank)
General_Coverage	(MetadataElement)	(blank)
Lifecycle_Contribute	RepositoryEntry.initialauthor	(blank)
Lifecycle_Publisher	(MetadataElement)	(blank)
Lifecycle_Version	(MetadataElement)	(blank)
Lifecycle_Status	(MetadataElement)	(blank)
Lifecycle_CreationDate	RepositoryEntry.creationdate	Course.timecreated
Technical_Format	CatalogEntry.Type	Resource.Type
Technical_Size	(MetadataElement)	(blank)
Technical_Location	RepositoryEntry.resourcename	Resource.reference
Rights_Copyright	(MetadataElement)	(blank)
Educational_Difficulty	(MetadataElement)	(blank)
Educational_EndUsers	(MetadataElement)	(blank)

Table 1. *MM_repository_entry* class - LMS systems mapping to the metamodel

Metamodel	**OLAT**	**Moodle**
Repository_entry	CatalogEntry.repoentry	Resource.id
Father	CatalogEntry.parentID	Resource.course

Table 2. *MM_catalog class* - LMS systems mapping to the metamodel

Metamodel	**OLAT**	**Moodle**
Id_user	UserImpl.id	CourseDisplay.userid
Id_repository_entry	PolicyImpl.oresource	CourseDisplay.course
Permission	PolicyImpl.permission	CourseDisplay.display

Table 3. *MM_permissions* class - LMS systems mapping to the metamodel

Metamodel	**OLAT**	**Moodle**
Source_id	ReferenceImpl.source_id	(blank)
Target_id	ReferenceImpl.target_id	(blank)

Table 4. *MM_references* class - LMS systems mapping to the metamodel

In the example we presented a part of Common Platform Independent Model for two Learning Management Systems: OLAT and Moodle; and a staged approach to Platform Independent Modelling that is based on a decomposition of the PIM into a common PIM and a target-specific PIM.

CONCLUSIONS

This contribution presents an original approach to the problem of integrating LMSs of different architectures. In particular, it introduces a staged decomposition of the PIM model into two submodels, the common PIM and the target-specific PIM. This refinement allows at the same time, to keep the benefit of the genericity of the PIM and to model functional specificities; given that such specificities otherwise would have been wiped out by an overly generic PIM. In the example we modelled a PIM of two LMS systems and showed how to map them to a generic common PIM.

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 difficulty to engineer generic courseware

REFERENCES

Brown, A., (2004). An introduction to Model Driven Archtiecture, IBM developer works. http://www-128.ibm.com/developerworks/rational/library/3100.html

Compose Project, (2004). An Overview to Domain Specific Language (DSL). http://compose.labri.fr/documentation/dsl/dsl_overview.php3

Grob, H. L., Bensberg, F., Dewanto, B.L., (2004), Model Driven Architecture (MDA): Integration and Model Reuse for Open Source eLearning Platforms, ELEED, University of Muenster. http://eleed.campussource.de/archive/1/81/

IEEE, (2002). Draft Standard for Learning Object Metadata, Institute of Electrical and Electronics Engineers. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

Moodle Project, (2006). Moodle Developper Documentation.
http://docs.moodle.org/en/Developer_documentation

Moreno, N., Romero, J.R., (2005). A MDA-based framework for building interoperable e-learning platforms, Recent Research Developments in Learning Technologies, Universidad de Málaga, Spain. http://www.formatex.org/micte2005/193.pdf

OLAT Project, (1999). University of Zurich, Switzerland. http://www.olat.org

OMG, (2001), A Proposal for and MDA Foundation Model, An ORMSC White Paper, Object Management Group. http://www.omg.org/docs/ormsc/05-04-01.pdf

OMG, (2003). MDA Guide version 1.0.1, Object Management Group. http://www.omg.org/docs/omg/03-06-01.pdf

SCORM, (2004). Advanced Distributed Learning, SCORM 2004 2nd Edition Documentation.
http://www.adlnet.gov/scorm/history/2004/documents.cfm

Siegel, J., (2005). Introduction to OMG's Unified Modeling Language, OMG. http://www.omg.org/gettingstarted/what_is_uml.htm

Wiley, J. (1996), CORBA – Fundamentals and Programming, John Wiley and Sons Publishing

Zuzana Bizoňová
INT Evrz, France
University of Žilina. Slovakia
Univerzitná 1, 01001 Žilina, Slovakia
Email: zubka@centrum.sk

Daniel Ranc
Institut National des Telecommunications
Ru Charles Fourier
91011 Evry Cedex, France
Email: daniel.ranc@int-evry.fr