MODEL DRIVEN LMS PLATFORM INTEGRATION

Zuzana BIZOŇOVÁ

University of Žilina, Faculty of Management Science and Informatics, Slovak Republic
e-mail: zubka@centrum.sk

Abstract

The growing choice of e-learning platform technologies leads to new difficulties. In particular, it becomes hard to manage or to compare available platforms and to re-use existing course material from different kinds of platforms because of technological discrepancy. This paper proposes a tentative solution of this difficulty - a common platform-independent framework that can be used to specify and classify existing or future Learning management systems (LMS). The proposed framework is based on OMG’s Model Driven Architecture.

Keywords: e-learning platform integration, learning management systems, model driven architecture,

1 INTRODUCTION

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 such as SCORM [1], IMS[2] or AICC[3]. 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 afore mentioned difficulties by using the Model Driven Architecture (MDA) [4] 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 LMS systems.

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 and OLAT. They serve as
examples of learning management systems with different technologies but similar functionalities. The section IV shows a possible solution of the proposed problem, introducing a staged approach to the Platform Independent Model of MDA. Finally, the last section describes concluding remarks and future work.

2 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 [5].

The MDA approach was proposed by OMG, the open standard organization supporting the well-known CORBA [9] and Unified Modeling Language (UML) [6]. The models in MDA [7] 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 behavior.

MDA uses the separation of concerns. It means 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 is 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 [8]:

- **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.

Platform in this case is a very relative term that depends on the point of view. Indeed, the same model can be a CORBA-specific PSM, because we decide to use CORBA [9] as middleware, but it is also a PIM with respect to operating system and hardware.
Models in the context of MDA Foundation are instances of metamodels and therefore consist of model elements and links between them. The elements, in turn, can hold attribute values which can be of primitive types, such as Integer, Boolean or String. [10]

A metamodel is a special kind of model that specifies the abstract syntax of a modeling language. It can be understood as the representation of the class of all models expressed in that language. Metamodels in the context of MDA are expressed using the Meta-object facility (MOF) [10].

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. They can see the commonalities and differences of systems at all levels.

Another advantage is to favor automatic transformation of the models to actual implementation code. The models serve as the basis for software architectures and the transformations can largely be automated.

Last but not least, MDA relies on open standards protecting projects against technology changes. Some Open standards that MDA uses for modeling [11]:

- the Unified Modeling Language (UML) – It is the suggested modeling notation for PIM/PSM but it is not compulsory to use it. Currently there is increasing interest in development of Domain Specific Language (DSL [12]).
- XML Metadata Interchange (XMI) [11] - 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.
- The Common Warehouse Metamodel (CWM) [11] - Standard interfaces that can be used to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments.
• 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.

3 TWO EXAMPLES OF OPEN SOURCE LMS SYSTEMS

Although most LMS have similar functionalities, platform integration and comparison of the systems might be difficult because of different web-based technologies used. Most of the open source systems are based on PHP, just a minority is 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 [13] 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 [16] 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 MDA FOR LMS FRAMEWORK

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 holistically 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](image)

It has to be noted that this staged approach is 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 of such an approach we can consider access rights for different kinds of users in OLAT and Moodle.

In OLAT [15] we recognize five kinds of users (Figure 3):

- **Guests**: they are anonymous users with restricted rights on functions. Guests can neither adjust their interface nor complete tests or participate in forums.
- **Users**: OLAT users may configure their interface according to their
preferences and start a course as participants.

- **Authors**: In addition to the rights of an OLAT user, authors can add, create and manage learning resources.
- **Group Administrator**: In addition to the rights of an OLAT user, members of this system group have the right to manage course overlapping study groups or rights of groups.
- **Administrators**: they are authorized to perform administrative tasks in the entire OLAT system.

![Access rights for different kinds of users in OLAT.](image)

**Figure 3** Access rights for different kinds of users in OLAT.

Roles in Moodle [16] are slightly different (Figure 4):

- **Guests**: they have read-only access, they can’t post discussions, submit assignments, contribute glossary entries and comments, receive scores and grades and similar tasks that need other than read-only access.
- **Students**: Students typically enroll in a course(s), submit assignments and receive scores and grades. Administrators and teachers determine how a student enrolls and what they can do in Moodle.
- **Teacher**: teacher has control over a specific course, can add, create and manage learning resources, organize students in the course, prepares tests and gives grades to students.
- **Administrators**: can do anything and go anywhere in the site. They are authorized to perform administrative tasks like creating new admins, removing admin rights etc.
Figure 4 Access rights for different kinds of users in 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** – a description of some particular Moodle feature. Capabilities are associated with roles. For example forum:replypost is a capability.
- **Permission** – value that is assigned for a capability for a particular role. For example allow or prevent.
- **Context** – space in the Moodle such as courses, activity modules etc.

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 metamodel in the proposed analysis. The metamodel integrates approaches from both systems. All users in both systems are allowed to do read-only activities, as can be seen on Figures 3 and 4. Therefore the parent class MM_Role contains read-only procedures that are inherited by all children classes: MM_Guest, MM_Student, MM_Teacher and MM_Admin. The children classes contain procedures that are allowed for a guest, a student, a teacher or an admin (Figure 5).

Thus in the Platform independent model specified for OLAT:

- Guest class inherits from MM_Guest class,
- User class inherits from MM_Student,
- Author class inherits from MM_Teacher and
- Group Administrator and Administrator inherit from MM_Admin.
Figure 5 Classes of Common Platform Independent Model. This model shows allowed activities for each class.

Though Moodle is not based on object-oriented principles, and PHP itself is not an object oriented language, there are some possibilities in the language to use the notion of objects. Therefore we will use classes even for this LMS. In Moodle:

- Guest class inherits from MM_Guest class,
- Student class inherits from MM_Student,
- Teacher class inherits from MM_Teacher and
- Administrator class inherits from MM_Admin.

The previously mentioned roles A and B of student can be modeled as child classes of Student class, Student::StudentA and Student::StudentB

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 Modeling that is based on a decomposition of the PIM into a common PIM and a target-specific PIM.

5 CONCLUSION

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 generosity of the PIM and to model functional specificities; given that such specificities otherwise would have been wiped out by an overly generic PIM.

We are currently at the early stage of the development and a lot of work has to be done yet. The next step is to integrate more LMS systems into our framework and to map those systems into the common metamodel. As we used SCORM metadata
standard in our approach, mapping will be much easier for SCORM compliant LMS systems.

We also plan to incorporate all other features and functionalities of the LMS systems into our common metamodel. 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
