Generalized Model for Interoperability of Data Based on Model Driven Architecture

Zuzana Bizonova

Telecom SudParis, Evry, France

Abstract. In the recent years, e-learning gained popularity among educational institutions as well as enterprises. As the result of that many commercial or open-source Learning Management Systems (LMS) were developed to manage online courses. 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 exchange pieces of information among those systems. Applications and their data become isolated what is a clear economical concern for the future of these technologies.
The present study describes a method, based on Model Driven Architecture (MDA), for integrating approaches of candidate LMS systems into a generalized architectural framework. The framework makes use of standards for description of data and metadata like learning materials (IEEE LOM, IEEE PAPI), student information (IMS LIP) or learning design (IMS LD). This platform-independent framework can be used for an automatic migration of data between various e-learning platforms.

Keywords: Interoperability, Model Driven Architecture, Learning Management Systems, Generalized Model.

1 Introduction

The sudden popularity of e-learning led to the development of a significant number of Learning Management Systems, either commercial or open source. Because of multiplicity of platforms and approaches used for various systems implementations, it has become increasingly difficult to manage interoperability of their data.

Their variety and growing number has become a true barrier for re-use of existing learning material. Creation of valuable interactive multimedia material requires a large commitment of time and resources. Due to the high costs associated with learning material development, a clear economic concern arises for the future of these technologies if the learning material and other kinds of data from LMS, such as student results and records, remain isolated with LMS applications.

Creation of valuable interactive multimedia material is demanding for time and ideas. Because of the cost of learning material development, if learning material stays isolated in applications, a clear economical concern arises for the future of these technologies. Not forgetting that other kinds of data from LMS, like student results and records, become isolated too.

The interoperability of e-learning systems has been intensively researched in recent years and several new standards have been created – for example SCORM [1]. It is a collection of standards and specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reusability of web-based learning content. Other examples of standards used in LMS systems are the IMS QTI [2] standard for tests and IMS LIP [3], for encapsulating learners' information and results.

However, most LMS systems have been created without regard to standards and therefore cannot be considered as part of an overall solution. But is it possible to achieve the goal of interoperability and data exchange even among LMS systems that are not based on standards?

The most serious obstacle to achieving this goal is that the various LMS systems have different architectures. Possible solutions would recognize these differences and try to find commonalities, and procedures to build bridges among the systems. This study aims to overcome this inherent difficulty with the current Learning Management Systems. In particular the issue of LMS interoperability among completely different architectures will be thoroughly examined.

For that purpose, we will define a new approach, based on the Model Driven Architecture [4] and using a detailed architectural analysis of candidate LMS systems which will produce different models of these systems. These models, of different levels of abstraction, will in turn be searched for commonalities and differences so that to identify unifying elements in their functionalities.

This new approach has resulted in a "three step method". This method defines a generalized model of a LMS, as well as mapping rules that will help to translate data from a LMS system to a generalized model and again to another LMS system.

The obtained generalized model should be based on standards and other generalizing ideas so that any other LMS can be added later to this framework.

In this article the theoretical concepts of our approach will be explained that will further be used to create a generalized model of LMS systems and mapping rules between candidate systems and the generalized system.

2 Method for the Creation of the Generalized Model

Our goal is to define a generalized model of LMS system consisting of features of other LMS systems that can be mapped into it. In this part we will introduce a three steps strategy to build up the final model. This model will then represent the foundation for the data interchange among systems.

As preamble to this three step method, an exhaustive functional analysis of candidate LMS systems has to be performed. This analysis builds the sum of all functionalities found in all candidate LMS systems, its outcome is the so-called *general functionality list (GFL)*

The three step method will now loop through all functionalities of the GFL to perform its tasks on each of them in turn. Let us suppose that a certain functionality F is now analyzed.

2.1 First Step

The goal of the first step is to search among standards at hand to select a suitable standardized functionality SF *supporting* the functionality F.

It is furthermore necessary to select the most general standard from the standards relevant for the chosen functionality F. The standards are used in this concept to enhance the resiliency of our analysis with respect to changes in the field of e-learning technology. Standards are generally supported and also based on the experience of many users. Furthermore we expect that if a change happens in the future of LMS systems, this change will be reflected at the standardization level; this change can then be back-ported to our model.

The standard function SF selected to support the functionality F should be so general that *it epitomizes F in as much LMS systems as possible, while being kept as specific as possible*:

The mapping between F and the supporting SF can be expressed as follows:

- *Strong* support - In this case SF includes strictly F, i.e. all of F is supported by SF.

$$\forall\ F_i \in LMS_i,\ \exists\ SF \in S_i \text{ so that } SF \supset F_i$$

- *Weak* support - In this case SF supports only partly F:

$$\forall\ F_i \in LMS_i,\ \exists\ SF \in S_i \text{ so that } SF \cap F_i \neq 0$$

- *No* support - In this case step one fails, when it cannot find any SF to support F, so that F needs to be applied "as is" to step two.

In any case the heuristic to select SF strives to minimize SF:

$$\max(SF \cap F) \text{ while } \min(SF)$$

2.2 Second Step (Reversed MDA paradigm)

Examining now from a top-down perspective the SF obtained in step 1, we may consider that each F of the general functionality list GFL corresponds to a particular flavour or *realization* of the standard functionality SF, as provided by the studied LMSs.

In the second step we are creating or enhancing the generalized model by continual integration of the functionality realizations as provided by candidate LMS systems (in this study Moodle [6], OLAT [7] and Claroline [8]).

This step is performed by examining the *weak support* cases of step one from the point of view of *missed functionality*. We consider in particular the cases where a missed functionality appears in at least two LMS candidates. If such a situation appears, we tag this functionality as *important* (i.e. a functionality shared by LMSs, but not covered by any standards) and select it to be integrated in the model:

$\exists\ F_1 \in LMS_1, \exists\ F_2 \in LMS_2, \exists!\ cf$ so that:

$cf \cap SF = 0$ and

$cf \subset F_1$ and

$cf \subset F_2$

where cf is a common functionality.

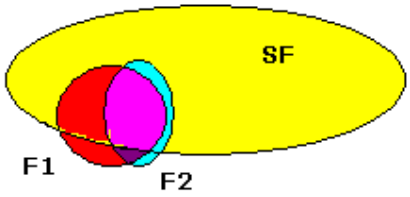

Fig. 1. Function F in LMS 1 is the red circle F1, the same function in LMS 2 is the blue circle. Their intersection is pink in the area of standard. The important missed functionality is the dark pink intersection of F1 and F2 – not covered by the standard but still in the intersection of definitions of functionalities of more than one LMS system.

For this purpose we use the Model Driven Architecture (MDA) that was previously described. Just to remind the reader of the concept, MDA is a way to organize and manage system architectures. The building of the system can be organized around a set of models by imposing series of transformations between them. The whole system creates an architectural framework of layers and transformations.
OMG defines three levels of abstraction (fig. 2) [12]:

- Platform Independent Model (PIM) – this model provides adequate functionalities, structure and behaviour 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.
- Implementation

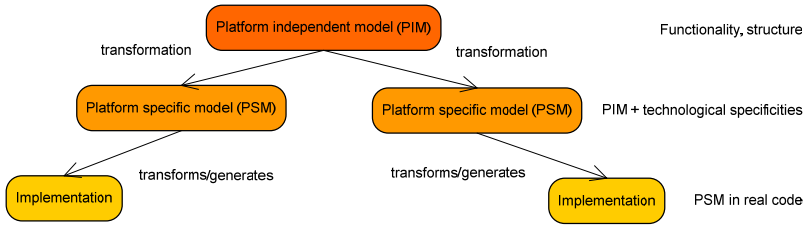

Fig. 2. MDA Concept

MDA principles are the background for the solution of the proposed problem with the LMS system integration.

Reversed MDA Paradigm

One of the reasons why to use MDA was that analyzing the system on different levels of abstraction helps us to understand the system better. The abstraction of the candidate LMS systems to the platform independent level can give us the necessary "look from above" to see the commonalities and differencies among various systems.
What we have at first (fig. 3) are the implementations of the LMS systems in frameworks of various technologies. These implementations can be abstracted to the PSM models and further to PIM models where technology used is irrelevant. At this level finally we can see the commonalities between various systems. Now that we got rid of implementation details of each LMS system, we are able to clearly see for example F1 and F2, it means how a certain functionality is realized in the architectures of various systems. We can describe and analyze them. The outcome of this step is a set of commonalities that can be further used to create (in case of *no support* from the first step) or enhance (in case of *weak support*) the General PIM by *important missed functionalities.*
This analysis also gives the foundation for the mapping rules from the functionality F1 in the LMS1 to the functionality F in the General PIM and from the functionality F in the General PIM to the F2 functionality in the LMS2.

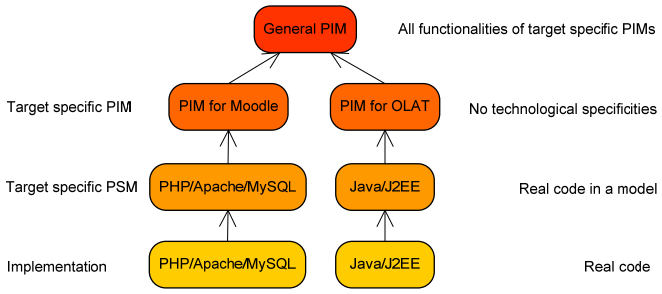

Fig. 3. Reversed MDA concept for creation of General PIM.

The previously described step repeats for each separate functionality in the system. Figure 4 shows the integration strategy to create the General PIM. For a certain functionality, PIM models of candidate systems are enhancing the General PIM until the model is saturated. Then we continue to enhance the generalized model with another functionality.
The systems used in our research were selected based on their variety of architecture, structure and technology used. It is to ensure that the General PIM contains a large combination of various functionality realizations and therefore many other systems can afterwards use this model.

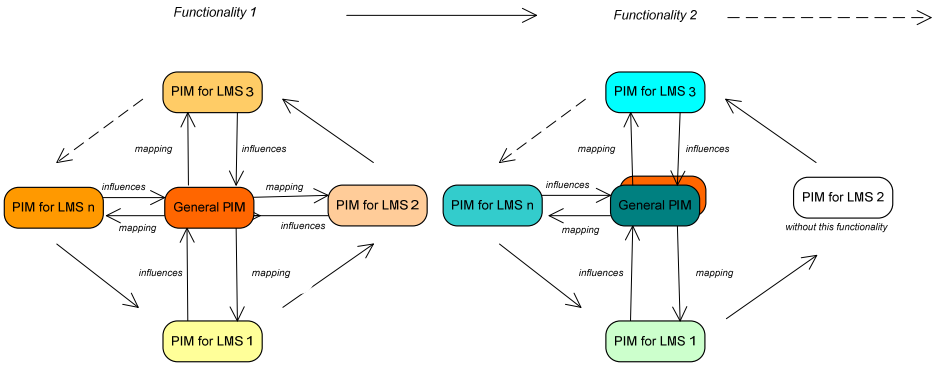

Fig. 4. Integration strategy

2.3 Third Step

The goal of the third step is to create **mapping rules** of the PIM models of candidate systems to the General PIM, and vice versa. Practically it means to create translation tables for data structures of a LMS system to the generalized system and the other way round.

Again, refering to the fig. 1 we can see that it is considerably easy to map functionality F1 of LMS 1 to the General PIM because General PIM should cover most of the F1 functionality (the intersection F1∩SF plus F1∩F2) and this part can be just “translated” to the terms of General PIM. The question is what to do with those values that are not covered by General PIM.

As we mentioned before, these missing features appear only in one of the candidate systems, therefore they are not incorporated in the General PIM. It means that we need to transfer them only in those cases when we transfer data from a system to the same system and we need to save also these extraordinary data. In this case we can use the class G_extra_metadata that will be introduced later in the sixth chapter. This class serves as a list of any definable attributes and their values that can be added to the main class G_repository_entry of any item in the model. This way we are able to create mapping rules for both the data that are incorporated in the General PIM as well for those that are not.

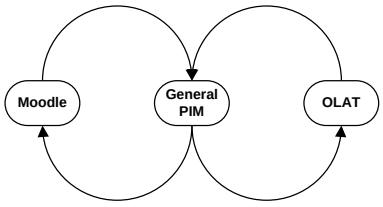

Fig. 5. Mapping rules – from Moodle to the General PIM, from General PIM to OLAT, and the other way round.

Systematic mapping between the candidate systems data and the data of the generalized system requires definition of mapping relations between entities of the target systems. Such relations, or mapping rules can have forms of 1:1 (*direct* mapping), 1:n (*divergent* mapping) or n:1 (*convergent* mapping). The set of mapping rules is constructed both ways: describing the transformations from the candidate system to the generalized model and the other way round. The direct mapping case is trivial. We simply "translate" one attribute to another one. The table will look like this:

General PIM	Candidate PIM
G_class.visible	Candidate_table.visible

Tab. 1. Relationship 1:1.

In the case where there is a convergent or divergent mapping, the situation has to be analyzed in more detail.

- simple convergent/divergent mapping

For example there can be a *redundancy* case where a value is mentioned twice in our generalized model or in the candidate model. Let us say there is just one type of name in the candidate table, but two types in the generalized model. The table can look like this then:

General PIM	Candidate PIM
G_class.shortname	**Candidate_table.name**
G_class.longname	**Candidate_table.name**

Tab. 2. Relationship n:1.

On the other hand 1:n relationship can be written in this manner:

Candidate PIM	General PIM
Candidate_table.name	G_class.shortname/ G_class.longname

Tab. 3. Relationship 1:n.

We do not tackle the situation of n:m relationship because of its great complexity. Such a relationship is however always composed of n:1 and 1:m relationships that can be implemented.

- complex convergent/divergent mapping

There can also be a complex value consisting of many objects that need to be combined. The actual combination law has to be defined manually (numbers: addition, strings: concatenation in simple cases). See an example on the table.

General PIM	Candidate PIM
G_class.keywords	Candidate_table.name+Candidate_table.author+ Candidate_table.subject

Tab. 4. A complex value.

3 Conclusion

The General PIM and the mapping rules represent the architectural framework that enables data interchange among LMS systems. Any piece of data can be translated to the General PIM with the mapping rules and translated again to the same piece of data in another system. As soon as any module is added into the General PIM and mapping rules of the LMS systems are written for it, we have gained the framework for the data transfer.

This generic approach can be used to enhance interoperability among systems that have not been created based on any standard as such case is not rare in the current technology enhanced learning environment. Such approach has been further developed and applied on three candidate systems in the dissertation thesis that was defended in September 2008 and the theoretical concepts have been proved by a demonstrator that successfully transformed various kinds of data between candidate systems, like tests and test questions, student results or even forums, chats and assignments [14].

References

1. Digital Think, SCORM: the e-learning Standard, (http://c-beta.digitalthink.com/dtfs/downloads/products_services/wp_standards.pdf)
2. IMS QTI, (http://www.imsglobal.org/question/)
3. IMS LIP, (http://www.imsglobal.org/profiles/index.html)
4. MDA, (http://www.omg.org/mda/)
5. Burgos, D., Tattersall, C., Dougiamas, M., Vogten, H., Koper, R.: Mapping IMS Learning Design and Moodle. A first understanding.
6. Moodle - Moodle project: Moodle Developper Documentation, (Nov 2006) (http://docs.moodle.org/en/Developer_documentation).
7. Gnägi, F.: 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. Claroline project – website (Nov 2007), (http://www.claroline.net/)
9. Mellor, S., Scott, K., Uhl, A., Weise, D.: MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley Professional, ISBN 0-201-78891-8
10. Kleppe, A., Warmer, J., Bast, W.: MDA Explained The Model Driven Architecture: Practice and Promise, Addison Wesley, ISBN 0-321-19442-X.
11. Miller, J, Mukerji, J.: MDA Guide Version 1.0.1, (2003) OMG,
12. A Proposal for an MDA Foundation Model, An ORMSC White Paper, 05-04-01
13. Soley, R.: Model Driven Architecture, OMG Staff Strategy, White Paper Draft 3.2 – (November 27, 2000).

14. Bizonova, Z. Model Driven E-learning Platform Integration, (Sep. 2008), (http://www.etwinning.sk/2008/MiddleI).