eleed
e-learning
& education

Issue 1
2005

Model Driven Architecture (MDA): Integration and Model Reuse for Open Source eLearning Platforms

Heinz Lothar Grob, Frank Bensberg, Blasius Lofi Dewanto

European Research Center for Information Systems (ERCIS)
University of Muenster
Leonardo-Campus 3, D-48149 Muenster, Germany

http://www.wi.uni-muenster.de/aw

grob@uni-muenster.de , bensberg@uni-muenster.de , dewanto@uni-muenster.de

urn:nbn:de:0009-5-817

Zusammenfassung

Die Open Source-Community bietet zahlreiche eLearning-Plattformen an: Learning Management-Systeme (LMS) sowie Learning Content-Systeme (LCS). Allgemeine Open-Source-Mediatoren, wie SourceForge, ObjectWeb, Apache und der eLearning-spezifische Mediator CampusSource ermöglichen eine einfache Suche nach eLearning-Softwareprodukten. Ein Vergleich unterschiedlicher Plattformen in Bezug auf ihre Funktionalitäten ist jedoch aufwändig. Beiträge aus der “eLearning Wikipedia” können kaum als Entscheidungsgrundlage genutzt werden, da sie schnell veraltet sind (1). Zudem fehlen derzeit Aktivitäten zur Integration von Open Source-eLearning-Plattformen, die oft über identische Funktionalitäten und Implementierungstechnologien verfügen. Solche Aktivitäten sind allerdings notwendig, da sowohl finanzielle und als auch personelle Ressourcen vieler Projekte stagnieren. Der vorliegende Beitrag skizziert eine mögliche Lösung der aufgezeigten Probleme mit Hilfe des MDA-Ansatzes. Er trägt zur Erkennung gemeinsamer Funktionalitäten bei und ermöglicht eine Integration von eLearning-Plattformen. Die Idee hierzu entstand anlässlich der zweiten CampusSource-Entwicklerkonferenz an der Westfälischen Wilhelms-Universität Münster am (27. August 2004).

Stichwörter: Model Driven Architecture (MDA), Learning Management System (LMS), Open Source (OS), Open University Support System (OpenUSS).

Abstract

Open Source (OS) community offers numerous eLearning platforms of both types: Learning Management Systems (LMS) and Learning Content Systems (LCS). General purpose OS intermediaries such as SourceForge, ObjectWeb, Apache or specialized intermediaries like CampusSource reduce the cost to locate such eLearning platforms. Still, it is impossible to directly compare the functionalities of those OS software products without performing detailed testing on each product. Some articles available from eLearning Wikipedia show comparisons between eLearning platforms which can help, but at the end they barely serve as documentation which are becoming out of date quickly (1). The absence of integration activities between OS eLearning platforms - which are sometimes quite similar in terms of functionalities and implementation technologies - is sometimes critical since most of the OS projects possess small financial and human resources. This paper shows a possible

solution for these barriers of OS eLearning platforms. We propose the Model Driven Architecture (MDA) concept to capture functionalities and to identify similarities between available OS eLearning platforms. This contribution evolved from a fruitful discussion at the 2nd CampusSource Developer Conference at the University of Muenster (27th August 2004).

Keywords: Model Driven Architecture (MDA), Learning Management System (LMS), Open Source (OS), Open University Support System (OpenUSS).

1. Introduction

Various Open Source eLearning platforms [1] available via intermediaries like CampusSource induce a severe economical dilemma: On the one hand, a broad spectrum of different products is available under permissive legal regulations of the OS domain. Consequently, potential adopters do not have to pay license fees or royalties. On the other hand, the OS domain offers poor documentation and integration between platforms, such that the transaction costs to find a product which meets the requirements are exorbitantly high. Practically speaking, the procurement process for an OS eLearning platform starts with installing a set of platforms and evaluating their features product by product.

A possible solution for this dilemma is the creation of comparative studies, which serve as foundation to compare available eLearning platforms in terms of functionalities and implementation technologies (2). Nevertheless, these studies show some severe deficiencies:

- OS software products have rapid development cycles and suffer from the symptom of lacking documentation. At best, the documentation state lies behind the product state. Consequently, comparative studies on eLearning platforms become quickly obsolete.
- Different authors have different views on identical functionalities, such that comparative studies only reflect some specific characteristics. In addition, comparative studies predominantly focus on enumerating static product criteria, which do not show the dynamic abilities to support specific eLearning processes. The creation and utilization of reference models is not common yet (3).

According to our experience, the majority of Open Source eLearning platforms shows following characteristics:

- Similar functionalities. Almost all LMS have collaborative mechanisms like discussion forums, chat tools and mailing lists. Though, it is difficult to compare these functionalities since product descriptions are based on different terminologies.
- Similar technologies. Most Open Source LMS are based on PHP, just a minority is implemented using Java and J2EE. Some LCS are directly integrated within the LMS itself, thus they use the same technology. Other LCS are completely independent and are mostly implemented with Java. [2]

Despite of some standardization activities like Open Knowledge Initiative (OKI) Sharable Content Object Reference Model (SCORM) and IMS Global Learning Consortium, there are no actions discernible, which try to integrate *similar* and *already available* eLearning

platforms. The result of a fictitious integration process of OS eLearning platforms would be a standardised platform, which covers a generic set of eLearning processes and business objects conforming to a generally accepted reference model.

The absence of integration is an interesting economical phenomenon, since most OS projects lack of financial and human resources. Certainly, there are some obvious barriers which prevent platform convergence:

- The installed base of eLearning platforms is high, so that it is cost intensive to migrate to a new platform.
- The knowledge on special implementation technologies (e. g. PHP, Java, C, C#, Linux, Windows) is available on the intra-project level. Learning other implementation technologies is costly.

Using Web Services as an integration technology would be inadequate, since Web Services can only be used to make different type of implementation technologies *interoperable* (4). The effort to *install*, *manage* and *maintain* each platform will still remain. In this area Web Services help platforms to become interoperable with others in order to support Service Oriented Architecture (SOA). They can use services offered by other platforms easily (5). Figure 1 shows how Web Services establish interoperability between four different OS LMS. Each platform can use functionalities and services offered by other platforms straightforwardly, regardless of different implementation technologies used by each platform. However it is almost unfeasible for an institution to install, manage and maintain four different LMS, implemented with different technologies to cover the needed eLearning infrastructure. In many cases the institution chooses one LMS, depending on the implementation knowledge of its human resources and extends it in case of adding new functionalities by its internal personnel.

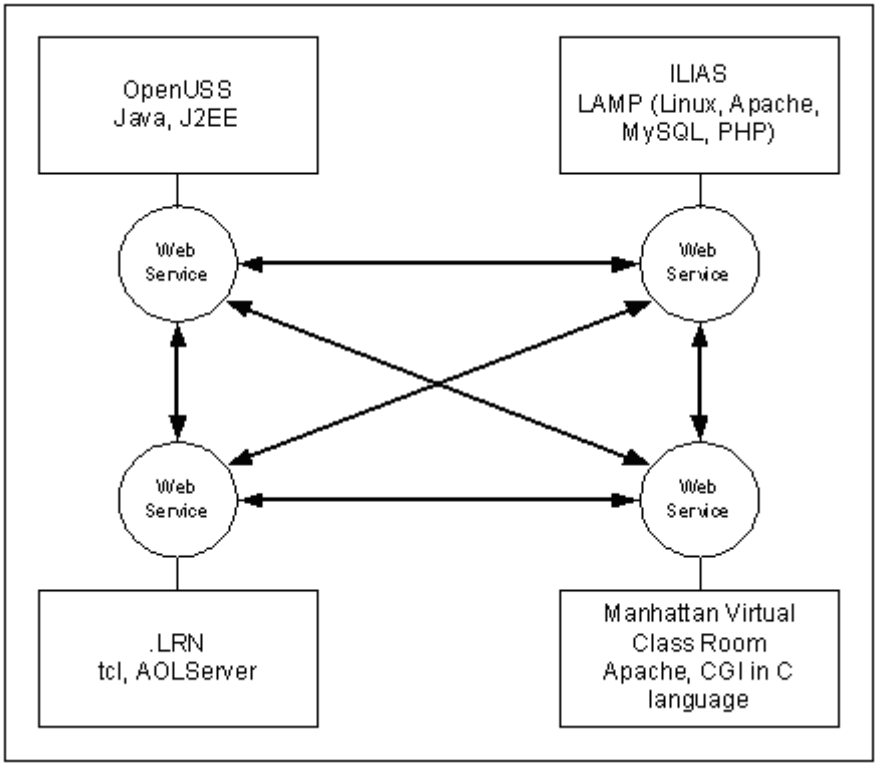

Figure 1 - Integration of Open Source LMS by Using Web Services

The mentioned integration problem is not an excuse to continue with the development of OS eLearning projects just like today. Following is a short summary of the requirements to all OS eLearning platform projects:

1. It must be easily possible to compare functionalities between all Open Source eLearning platforms (LMS and LCS).
2. The reuse of functionalities between eLearning platforms must be possible.
3. The co-existence of heterogeneous implementation technologies has to be accepted, since institutions have made technology specific investments.
4. To administrate and to optimize different platforms with different technologies is cost intensive. Therefore it is preferable to install one platform based on a homogeneous technology.

To fulfil these requirements, we propose the use of the *Model Driven Architecture* (MDA) concept in the domain of Open Source eLearning platforms. The next section gives a short introduction to MDA. Subsequently, section 3 shows a possible solution to the identified requirements using MDA. Afterwards, section 4 will present a real world example of how this concept can be implemented.

2. Basics of MDA Concept

In general, MDA aims to separate business, application or domain logic (*Platform Independent Model* = PIM) from the underlying implementation technology (*Platform Specific Model* = PSM and *Implementation*) (6)(7)(8)(9)(10)(11). The separation of PIM, PSM and implementation conforms to the most important rule in information systems development, the *separation of concerns* (12)(13). Additionally, using PIM allows us to abstract from the implementation technology, such that information system development reaches a higher level of abstraction. This augmentation in abstraction is comparable to the evolution of programming languages, from machine language to assembler to procedure and module oriented language and today to object oriented language.

The idea to separate platform independent and specific models is not new. For instance, the Architecture of Integrated Information Systems (ARIS), which has gained acceptance in Enterprise Resource Planning (ERP) systems, has already proposed the same concept. A major drawback of ARIS concept is the lack of formal transformation mechanisms, which ensure the automatic translation of platform independent models to platform specific models (14). In contrast to this, MDA explicitly uses *transformation rules* to translate models. These mechanisms accomplish transformation from PIM to PSM and to implementation. Figure 2 shows a comparison between ARIS and MDA concept.

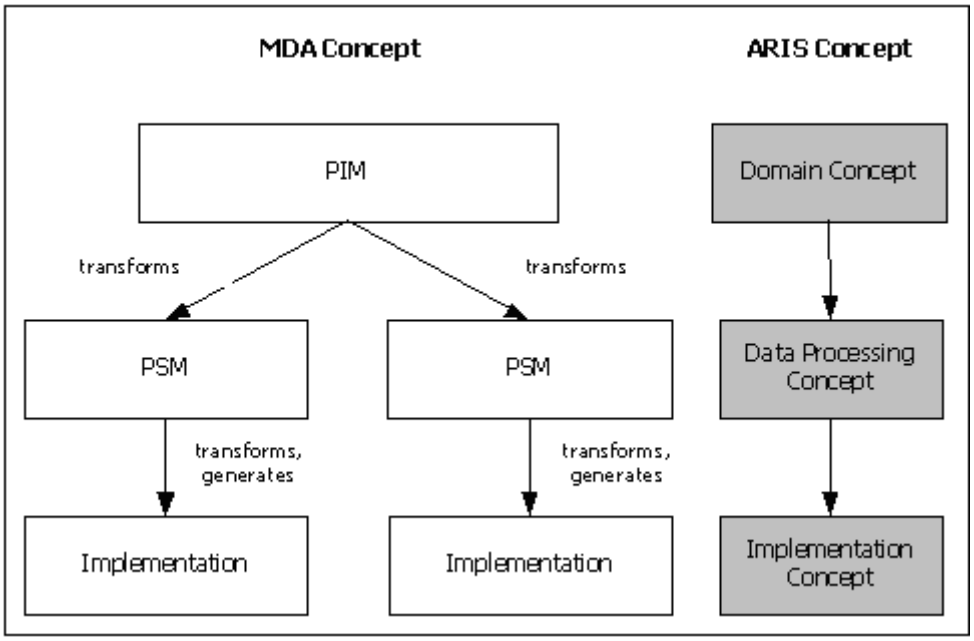

Figure 2 - Comparison of MDA and ARIS

For the PIM and/or PSM modeling purpose, the Object Management Group (OMG), creator of MDA, suggests using the Unified Modeling Language (UML) (15), which originally also comes from this institution. The use of UML is not a must and currently there is increasing interest in development of the so called Domain Specific Language (DSL) (16).

To have the ability to define DSL or to use UML for special purposes, the concepts *model*, *metamodel* and *meta-metamodel* play a fundamental role. Also transformation rules in MDA are based upon the same concept (7). For this intention OMG defines the so called OMG Meta-Object Facility (MOF) which includes a four layer architecture. Figure 3 presents MOF four-layer architecture with an example.

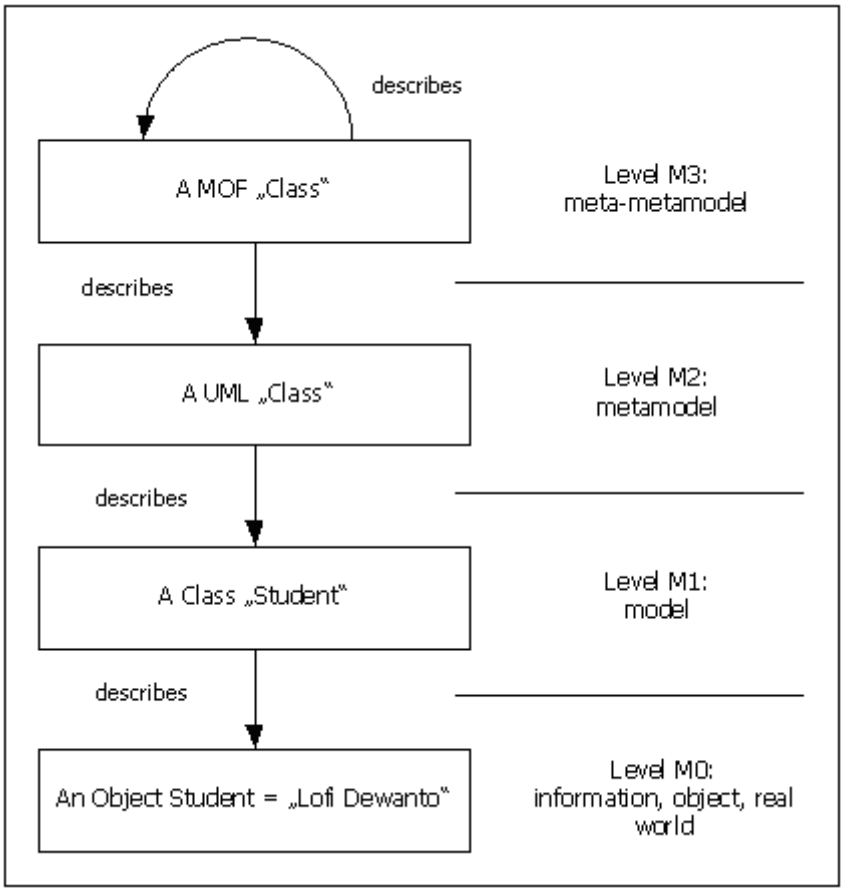

Figure 3 - MOF Four Layer Architecture

The lowest level M0 shows a real world object of a Student who has the name ”Lofi Dewanto”. To be able to describe the Student object ”Lofi Dewanto”, we need to declare a Student class at the level of M1. This Student class comprises all objects which have similar characteristics. At level M1 we use UML as modeling language. This is the typical approach of modeling and implementing object-oriented information systems today.

The level M2 describes the metamodel of UML which serves as a tool to check the correctness of the model semantic developed in the level M1. An example of this would be:

- A Class can have an Association only with other Classes.
- A Class can extend another Class (Generalization).

At the level M2 UML metamodel offers a MetaClass ”Class” modeling element to describe the Class at the lower level M1.

The level M3, the meta-metamodel layer, is the highest layer. UML metamodel from level M2 needs a description and this is where MOF plays its role. In our example, MOF ”Class” describes UML metamodel ”Class”. Since MOF describes itself, it does not require further metamodels.

MOF has a very small set of constructs to describe other models below the M3 level. The most important constructs are *Classes*, *Associations*, *DataTypes* and *Packages* (17). Using MOF we can define metamodels of any DSL, just like how MOF describes UML metamodel. Therefore it is clear that MDA does not depend on UML as modeling language. It is possible to create your own DSL (metamodels), which will be described by MOF. Thus, MDA is very flexible and powerful.

One important aspect in MDA is the availability of *transformation rules* – sometimes also called *mappings* – which will transform one model into another model or source code. Currently, OMG works on a transformation rules language, the so called QVT (Query/-Views/-Transformations) standard (18). Since QVT is not available yet, one can use JMI (Java Metadata Interface) for transformations using Java programming language (19). It is also possible to employ XSLT (Extensible Stylesheet Language Transformation), since all models and metamodels are available in the form of XML documents, the so called XMI (XML Metadata Interchange) documents (20).

Furthermore, the *openness* of MDA concept needs to be emphasized. Currently, there are a set of Open Source implementations for MDA tools, which are essential for MDA diffusion within the developer community. [3]

3. MDA for eLearning Domain

eLearning is considered as a specific information systems domain among others - like eBusiness, eGovernment, eHealth, eFinancial. In these domains, the MDA concept already has proven its value for software engineering (21). Also in the domain of eLearning, the MDA concept is not a brand new topic (22). Rather the question is whether MDA is able to solve our Open Source integration dilemma in the eLearning domain. Consequently, we have to evaluate the MDA concept against our requirements mentioned in section 1. We start with a scenario, which integrates different OS LMS under the MDA umbrella. Figure 4 shows a diagram of MDA concept with one common PIM mapped to different OS LMS.

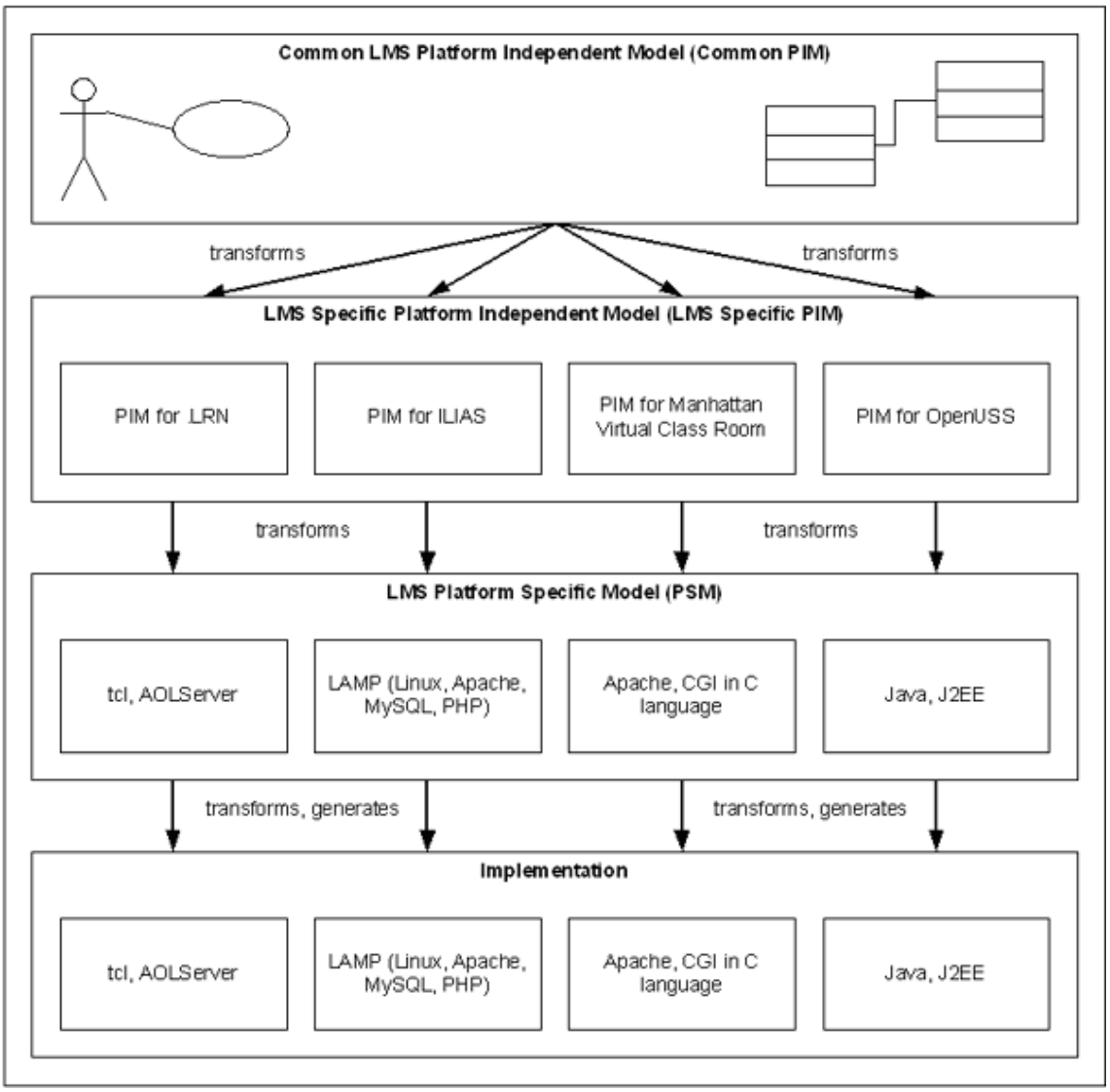

Figure 4 - Applying MDA to Open Source LMS

Following is a step-by-step solution to apply MDA to OS LMS:

- Each eLearning platform should add a PIM, the so called *LMS specific PIM*, into its development cycle. In our example we will have PIMs for OpenUSS, ILIAS, .LRN and Manhattan Virtual Classroom. By having a PIM for each platform, it is easy to compare functionalities between platforms. We propose to use UML as modeling language, because UML is widespread and has a mechanism to let us create our own UML-based DSL by using *profile* and *metamodel extension*. Furthermore, UML has a very strong tool support. [4]
- After each eLearning platform has its own PIM, they should be systematically compared and refactored. Identical PIMs will be integrated into a *Common PIM*. The more we can group the existing PIMs into this Common PIM, the more we can reuse them. In a perfect world we would only have one Common PIM at the end – this means

that all LMS will have same functionalities and structures. They only differ in implementation technologies.

- In order to transform the Common PIM into the LMS Specific PIM, we use transformation rules. Afterwards we transform the LMS Specific PIM into a PSM and at the end into the implementation platform.

Certainly, the creation of a Common PIM for the domain of eLearning is a visionary goal, because it is impossible to calibrate the functionalities and structure of all the LMS within a short time. Nevertheless, this approach shows some other benefits:

- We will have *a reusable Common PIM*, which can be used as basis system (”eLearning Engine”). In the long run, we may have only *one Common PIM* for all LMS and *no LMS specific PIM*. Consequently, we achieve cost effectiveness (see requirements #2 and #4 in section 1).
- It is possible to *easily compare the functionalities and structures*. Also the documentation of the platforms will be synchronised with the systems as the PIM will be directly integrated into the development cycle of the LMS (see requirement #1 in section 1).
- The actors of the OS projects can still use and maintain their knowledge in specific implementation technologies. Consequently, *investments into knowledge and human resources are preserved* (see requirement #3 in section 1).
- To move to another implementation technology we just change the *transformation rules* and re-implement some parts of the LMS using the new technology.

In order to substantiate the benefits and feasibility of our innovative approach, we provide an example for an OS LMS.

4. Example

To prove the potentials of MDA in the real world, we use Open University Support System (OpenUSS), which is an Open Source LMS based on Java2 Enterprise Edition (J2EE). OpenUSS serves as a simple step-by-step example according to the solution we already presented above (23)(24)(25).

As development infrastructure, we use the EJOSA concept. This acronym stands for Enterprise Java Open Source Architecture (EJOSA). EJOSA was invented to achieve the separation of concerns, because the development of OpenUSS was getting very complex (see Figure 5) (26)(27). EJOSA also gives some best practices, such as a useful directory structure for the components, separation of layers and also a straight and easy to understand development process in developing multi-tier applications. It also plays a support role for component development within OpenUSS, so that many developers can easily add individual components to OpenUSS (28)(29)(30)(31). Since EJOSA supports MDA, it simplifies refactoring OpenUSS. This conceptual basis is not common for the majority of Open Source eLearning platforms. Most of them were implemented without separation of concerns in mind.

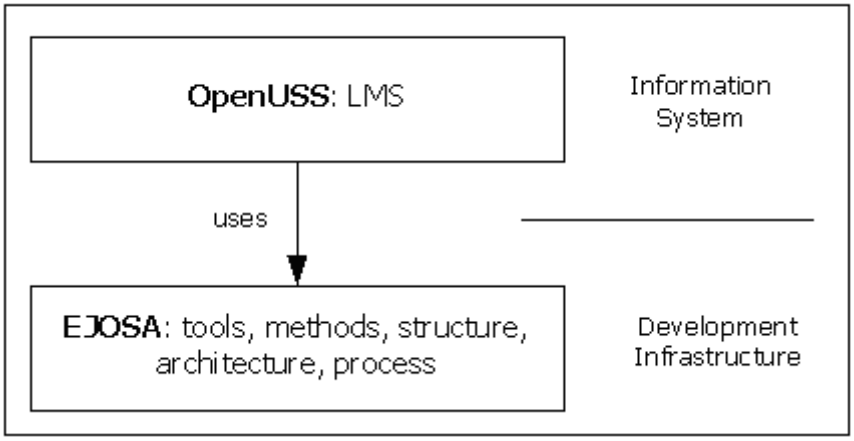

Figure 5 - Separation of Concerns: EJOSA as Infrastructure for OpenUSS

MDA support in EJOSA is realised by using AndroMDA (32). This is an Open Source, Java-based tool supporting model driven development. It has an open and pluggable design, such that many of its components can be extended easily. The core concept of AndroMDA is the use of so called *cartridges*. A cartridge describes the transformation rules from PIM over PSM to implementation code (see Figure 6).

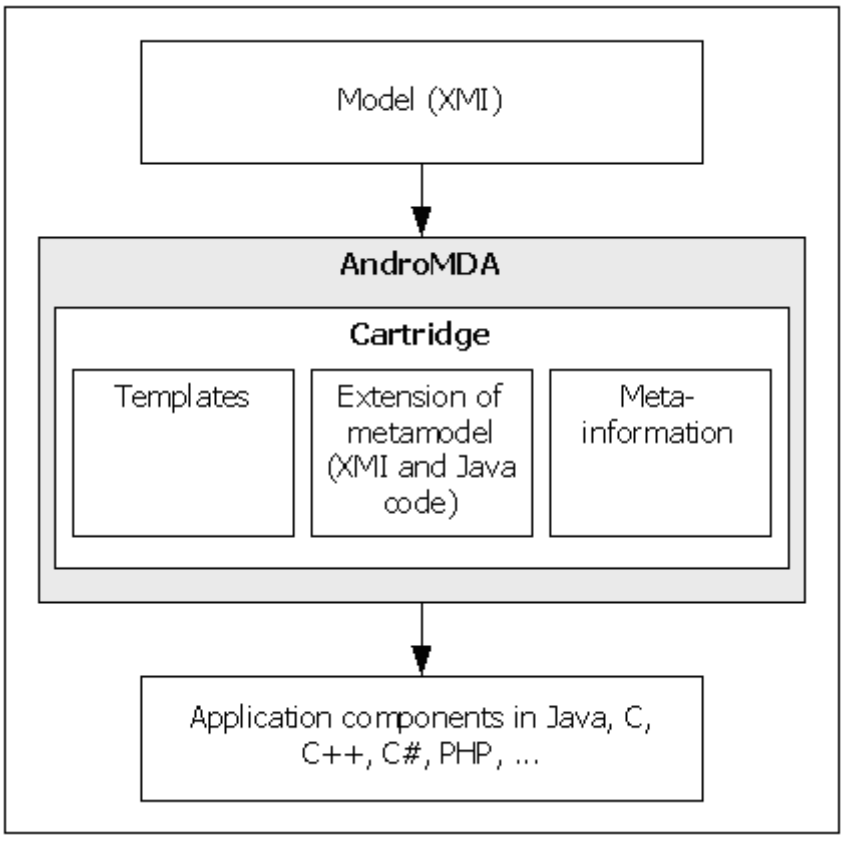

Figure 6 - AndroMDA Concept

AndroMDA brings some cartridges with it. To scope with the EJOSA structure we extended and customized some of the cartridges. Additionally we implemented some new cartridges, which are important for EJOSA. Table 1 shows the cartridges in EJOSA which can be divided into the *specification* (Application Programming Interface – API) and different layers of implementation like *business* layer implementation and *presentation* layer implementation (27).

Type	Cartridges
Model	Language Transformation
Specification (API)	EJB, Remote, Independent
Business (implementation)	EJB, EJB - Hibernate
Presentation (implementation)	Enhydra - XMLC

Table 1 - AndroMDA-Cartridges for EJOSA

Following steps are necessary to use AndroMDA as MDA tool:

- The PIM has to be modeled with an UML tool like PoseidonUML Community Edition, which is free to use but not Open Source (33).
- The PIM has to be marked with *stereotypes* and *tagged values*, depending on the cartridge to be used.
- The PIM is exported into a XMI document, so that AndroMDA can use this XMI document as an input. AndroMDA will create all artefacts (source codes, documents, models) driven by stereotypes.
- Compile the PIM to create source codes, other models and other documents. The result depends on the utilized cartridge (34).

For simplicity we will focus on the Student component described in Figure 7. Following steps are necessary to implement our MDA approach:

Step 1: Creation of an LMS specific PIM, in this case a PIM for OpenUSS.

Figure 7 shows the PIM for OpenUSS. In this PIM you can see that the Student class extends the Person class. Both have the type of <<Entity>>, which will drive the generation process of AndroMDA.

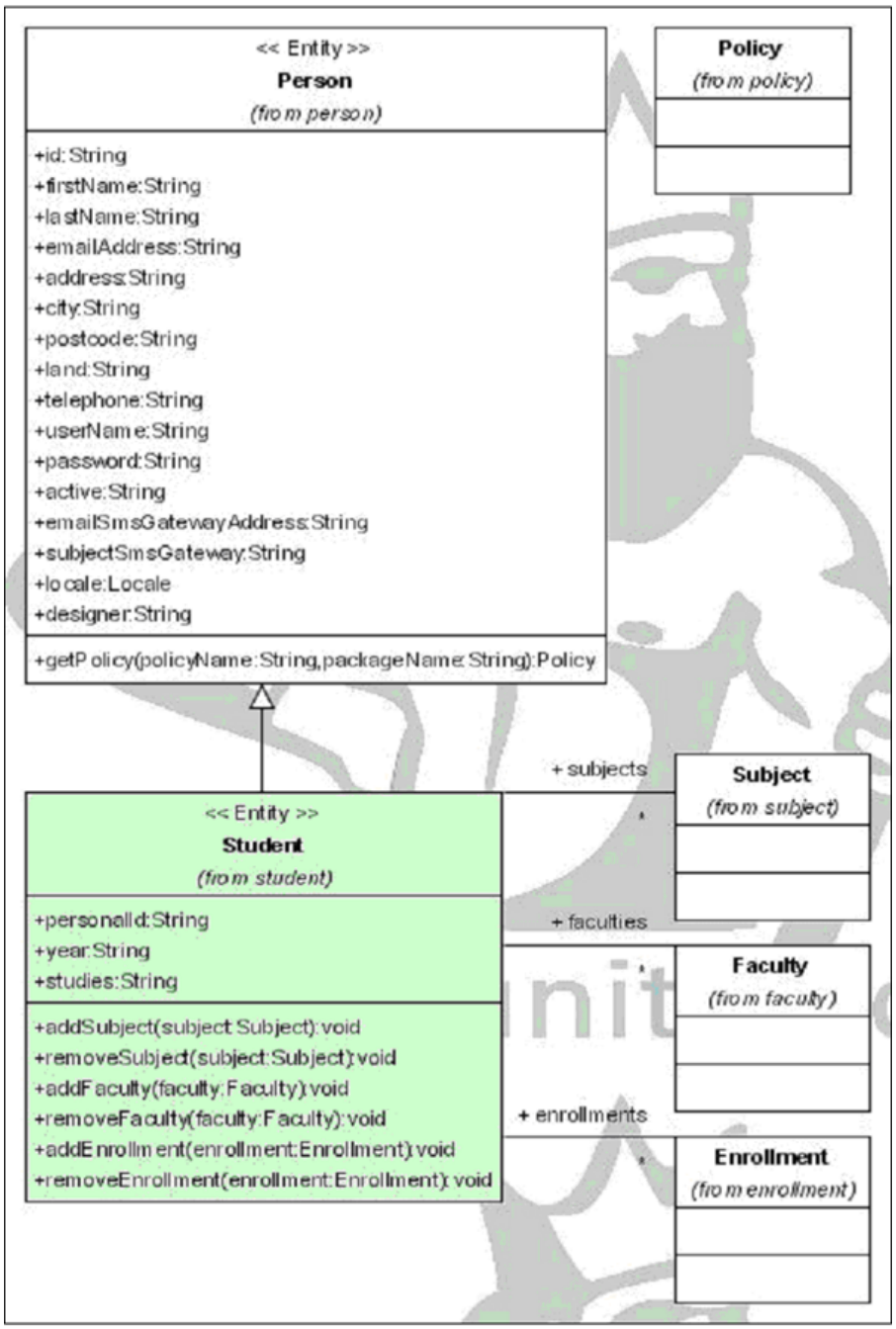

Figure 7 - PIM for OpenUSS: Student Class

Step 2: Usage of AndroMDA for code generation.

The code must be the same as the already available code. The AndroMDA cartridge can be customized to get the same code. Code generated by AndroMDA using the *ejosa-specification-remote* cartridge will look like below. This example is identical with the current code in the OpenUSS specification:

Person.java:

```
package org.openuss.business.foundation.person;
public interface Person {
public java.lang.String getId() throws java.rmi.RemoteException;
public void setId(java.lang.String newValue) throws
java.rmi.RemoteException;
public java.lang.String getFirstName() throws
java.rmi.RemoteException;
…
}
```

Student.java:

```
package org.openuss.business.foundation.student;
public interface Student extends
org.openuss.business.foundation.person.Person {
public java.lang.String getPersonalId() throws
java.rmi.RemoteException;
public void setPersonalId(java.lang.String newValue)
throws java.rmi.RemoteException;
public java.lang.String getYear() throws java.rmi.RemoteException;
…
}
```

We can now remove the current specification code as we always regenerate them from the PIM for OpenUSS automatically. In the first iteration we can replace the specification (API) with the model. Afterwards we can additionally generate almost all of the *business* and *presentation* implementation layer. This process should be done iteratively.

Step 3: Creation of the Common PIM and refactoring the LMS specific PIM.

After we have all the PIMs for each LMS e.g. PIM for OpenUSS, PIM for ILIAS, PIM for Manhattan Virtual Class Room and PIM for .LRN – in this paper we only show how to create the PIM for OpenUSS (Step 1 – 2 above), for any other platforms the process looks similar – we can discuss to move some of the attributes, methods or classes into its own place. Let's say that we agree with all LMS on a new class PersonCommonEntity, which has id, lastName and firstName as attributes.

In the OpenUSS PIM we need to delete all those attributes and change the stereotype of the Person class from <<Entity>> to <<PersonCommonEntity>> (refactoring the PIM for OpenUSS). We also need to customize the cartridge, such that AndroMDA will also take care of the <<PersonCommonEntity>> stereotype. All the classes which have this stereotype will extend automatically from a PersonCommonEntity class. [5]

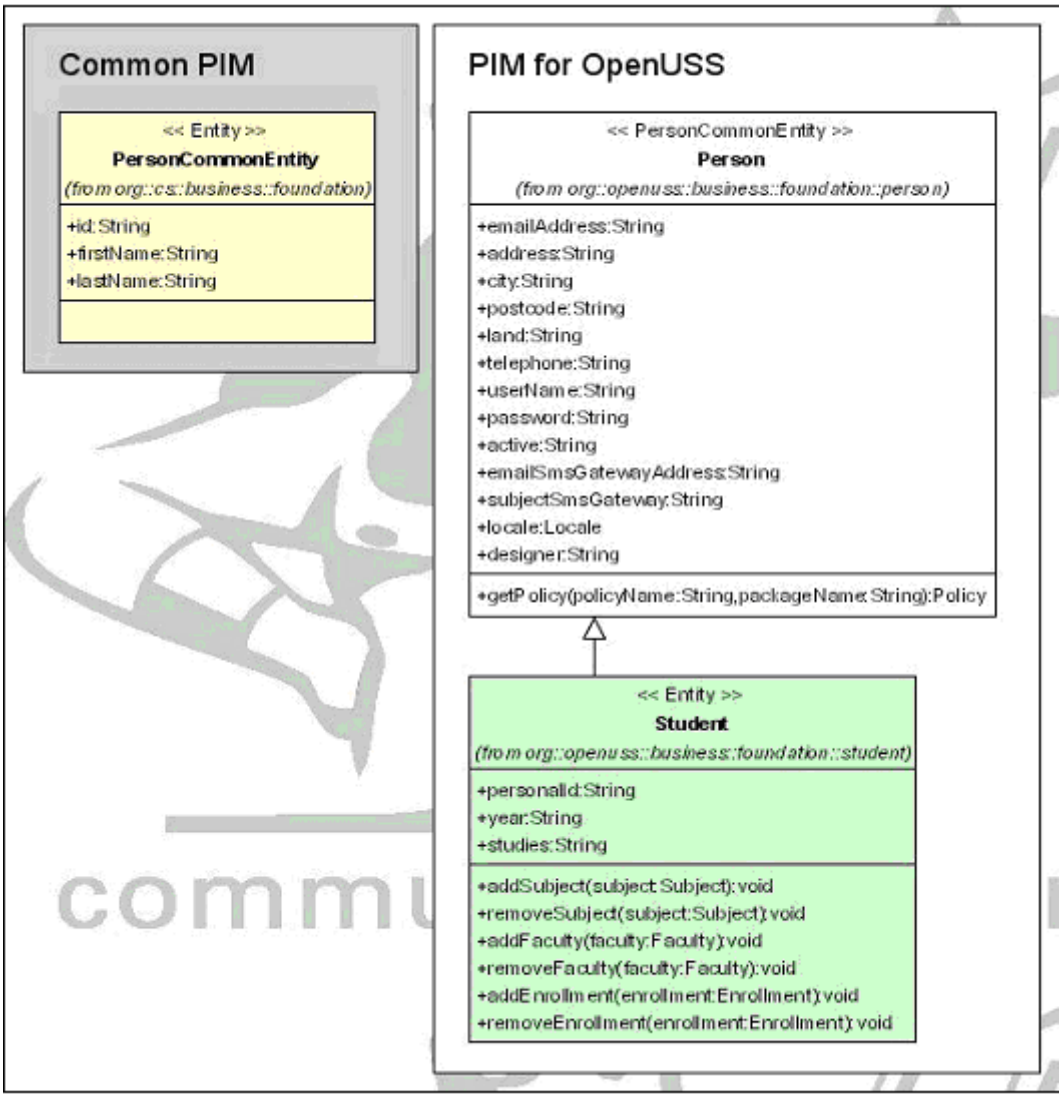

Figure 8 - Common PIM: PersonCommonEntity and updated PIM for OpenUSS

Step 4: From now on, forward engineering is carried out, using the ”model” as the ”first class citizen” source.

Changes in PersonCommonEntity will also directly change the implementation of the Person class in OpenUSS since it uses the <<PersonCommonEntity>> stereotype. Following code shows the inheritance relationship between Person and PersonCommonEntity, which will be automatically regenerated from the Common PIM and the PIM for OpenUSS each time those PIMs are modified.

PersonCommonEntity.java:

```
package org.cs.business.foundation;
public interface PersonCommonEntity {
public java.lang.String getId() throws java.rmi.RemoteException;
public void setId(java.lang.String newValue) throws
java.rmi.RemoteException;
```

```
public java.lang.String getFirstName() throws
java.rmi.RemoteException;
…
}
```

Person.java:

```
package org.openuss.business.foundation.person;
public interface Person extends
org.cs.business.foundation.PersonCommonEntity {
…
}
```

5. Summary

At the moment it is difficult to compare functionalities of available Open Source eLearning platforms, such that the search costs to find a product which fits the requirements are exorbitantly high. At the same time, the absence of integration activities between Open Source eLearning platforms is critical since most of the Open Source projects exhibit declining financial and human resources.

Using Web Services for the integration of technologically heterogeneous Open Source eLearning platforms is a pragmatic approach. However, in the long run it is better to pursue the goal of a single Open Source LMS, which is written in one language, so that Open Source developers can contribute on the same product instead of different Open Source products. The problem with this approach is that the choice of programming language depends on many factors. At the moment surely Java is the choice but many developers opt for PHP as a scripting language since it is pure Open Source, it has a big community and almost all system administrators know how to manage it. Also the knowledge already invested in those languages and technologies must not be neglected.

Our solution based on MDA will help to reuse the models without becoming dependent on specific technologies. The automatic transformation and generation from models also assure the synchronization with the actual implementations and documentations, so that the models are the ”source codes” of development. This enables all developers to reuse reference models, customize and refine models to fulfil their own requirements, and will ensure a high degree of TCO effectiveness (35). With the MDA support of OpenUSS through EJOSA, we present the first milestone to create eLearning components based on platform independent models. Visionary speaking, this milestone serves as a foundation for an eLearning engine, which provides basic services and can be reused within a broader spectrum of use cases.

6. Acknowledgements

The authors would like to thank all developers of CampusSource for their fruitful input and discussion on MDA during the 2nd CampusSource Developer Conference at the University of Muenster (27th August 2004).

7. References

(1) Wikipedia – the free encyclopedia, ”E-Learning,” http://en.wikipedia.org/wiki/E-learning (current July 2004).

(2) EduTools, LMS Comparison Articles Website, http://www.edutools.info/course (current September 2004).

(3) E. E. Doberkat, et al., Anforderungen an eine eLearning-Plattform – Innovation und Integration, Dortmund, 2002.

(4) M. Gehrke, M. Meyer, W. Schäfer, ”Eine Rahmenarchitektur für verteilte Lehr- und Lernsysteme,” http://www.campussource.de/projekte/docs/rahmenarchitektur.pdf (current July 2004).

(5) G. Vossen, P. Westerkamp, ”E-Learning as a Web Service,” Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS’03).

(6) OMG, MDA Website, http://www.omg.org/mda (current September 2004).

(7) A. Kleppe, J. Warmer, W. Bast, MDA Explained, The Model Driven Architecture: Practice and Promise, Boston 2003.

(8) S. J. Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled, Principles of Model-Driven Architecture, Boston et al. 2004.

(9) S. J. Mellor, K. Scott, A. Uhl, D. Weise, ”Model-Driven Architecture,” Advances in Object-Oriented Information Systems, OOIS 2002 Workshops, LNCS 2426, Berlin 2002, page 290-297.

(10) D. S. Frankel, Model Driven Architecture, Applying MDA to Enterprise Computing, Indianapolis et al. 2003.

(11) M. Belaunde, et al., ”MDA Guide Version 1.0.1,” http://www.omg.org/docs/omg/03-06-01.pdf (current July 2004).

(12) N. Habra, ”Separation of Concerns in Software Engineering Education,” http://researchweb.watson.ibm.com/hyperspace/workshops/icse2001/Papers/habra.pdf (current July 2004).

(13) E. L. A. Baniassad, G. C. Murphy, C. Schwanninger, ”Determining the ”Why” of Concerns,” http://www.research.ibm.com/hyperspace/workshops/icse2001/Papers/baniassad.pdf (current July 2004).

(14) P. Fettke, P. Loos, ”Model Driven Architecture (MDA),” Wirtschaftsinformatik, 45, 2003, page 555-559.

(15) OMG, UML Website, http://www.omg.org/uml (current September 2004).

(16) COMPOSE Project, ”An Overview to Domain Specific Language (DSL),” http://compose.labri.fr/documentation/dsl/dsl_overview.php3 (current September 2004).

(17) OMG, ”Meta Object Facility (MOF) Specification,” http://www.omg.org/docs/formal/02-04-03.pdf (current July 2004).

(18) OMG, ”QVT Request for Proposal and a Revised Submission,” http://www.omg.org/docs/ad/02-04-10.pdf and http://www.omg.org/docs/ad/03-08-08.pdf (current July 2004).

(19) Sun Microsystems, JMI Specification Website, http://java.sun.com/products/jmi/index.jsp (current July 2004).

(20) UMT-QVT, Open Source MDA ToolUsing JMI and XSLT as Transformation for Models Website, http://umt-qvt.sourceforge.net (current July 2004).

(21) OMG, ”Success Stories about MDA,” http://www.omg.org/mda/products_success.htm (current July 2004).

(22) H. Wang, D. Zhang, ”MDA-based Development of E-Learning System,” Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC’03).

(23) B. L. Dewanto, ”Learning Java Programming Language with Open Source Products and Technologies,” http://edu.-netbeans.-org-/-support/oss.html (current July 2003).

(24) H. L. Grob, Informationsverarbeitung in der Hochschullehre, Working Paper, Computer Assisted Learning and Computer Assisted Teaching, No. 25, Muenster, 2003.

(25) ObjectWeb, ”JOnAS and Enhydra pass OpenUSS entrance exam,” http://www.objectweb.org/wws/d_read/marketing/public/SS_OpenUSS_2p.pdf (current January 2004).

(26) B. L. Dewanto, ”Enterprise Java Open Source Architecture, User Manual,” http://prdownloads.sourceforge.net/ejosa/ejosa-revo2.2-doc.pdf?download (current January 2005).

(27) B. L. Dewanto, ”The Enterprise Java Open Source Architecture Template - A Beginners' Way to J2EE Application Development - Part I,” http://www.jaxmagazine.com/itr/online_artikel/psecom,id,653,nodeid,147.html (current January 2005).

(28) F. Bensberg and B. L. Dewanto, ”Entwurfsmuster bei der Implementierung von OpenUSS Teil 1: Mustergültig,” JavaMagazin Ausgabe 12, 2001.

(29) F. Bensberg and B. L. Dewanto, ”Entwurfsmuster bei der Implementierung von OpenUSS Teil 2: BLOB: Big and Beautiful,” JavaMagazin Ausgabe 02, 2002.

(30) F. Bensberg and B. L. Dewanto, ”Entwurfsmuster bei der Implementierung von OpenUSS Teil 3: Divide et impera!,” JavaMagazin Ausgabe 06, 2002.

(31) H. L. Grob, F. Bensberg and B. L. Dewanto, ”Developing, Deploying, Using and Evaluating an Open Source Learning Management System,” Journal of Computing and Information Technology (CIT), Vol. 12 (2004), No. 2, pp. 127-134.

(32) AndroMDA, Project Website, http://www.andromda.org (current September 2004).

(33) Gentleware, PoseidonUML Community Edition Website, http://www.gentleware.com (current September 2004).

(34) AndroMDA, AndroMDA Documentation - How to Model PIM - Website, http://www.andromda.org/modeling.html (current January 2005).

(35) F. Bensberg and B. L. Dewanto, ”TCO VO-FI for eLearning Platforms,” Poster Abstracts of the 25th International Conference on Information Technology Interfaces (ITI 03), SRCE, Zagreb (Croatia), 2003, pp. 9-12.

---

[1] A portfolio of Open Source LMS is provided by the intermediary CampusSource. Further examples are uPortal, CHEF, .LRN, Manhattan Virtual Classroom and Moodle. A good link list of LMS and LCS can be found at: http://www.elearningworkshops.com/--modules.php?-name=Web_Links&-l_op=viewlink-&cid=27 [01.09.2004].

[2] Javanti and FSL are some examples of OS LCS.

[3] A valuable discussion about this topic can be found at: http://www.theserverside.com/news/thread.tss?thread_id=26853 [01.09.2004].

[4] A valuable discussion about UML, UML-based DSL, MOF-based DSL vs. proprietary DSL can be found at: http://www.theserverside.com/news/thread.tss?thread_id=30488 [18.01.2005].

[5] You also can use inheritance in this case but it is better to use stereotype since we can model PersonCommonEntity from the Common PIM independently from the Person class within the PIM for OpenUSS.