FROM THE PLATFORM INDEPENDENT MODEL (PIM) TO THE FINAL CODE MODEL (FCM) ACCORDING TO THE MODEL DRIVEN ARCHITECTURE (MDA)

Miguel Ángel Sánchez Vidales
Pontifical University of Salamanca
Salamanca, Spain

Ana Fermoso García
Pontifical University of Salamanca
Salamanca, Spain

Luis Joyanes Aguilar
Pontifical University of Salamanca
Madrid, Spain

ABSTRACT
OMG (Object Management Group) has defined the MDA (Model Driven Architecture) specification trying to solve the actual problems of systems integration and interoperability. The main goal of this work is to introduce some basic concepts of MDA and show that is possible to define methods to transform a Platform Independent Model (PIM) to different Final Code Models (FCM), this is, the same high level model converted to different low-level models each one in different programming languages.

KEYWORDS
MDA, MOF, UML, PIM, PSM, Modeling.

1. INTRODUCTION
Nowadays it is difficult to model and implement software applications due to the complexity and variety of the technological architectures implied in its development and execution: hardware platforms, operating systems, middleware, networks, databases, etc. Nevertheless, companies demand more quality, productivity and integration. To solve this problem the MDA specification [OMG, 2003] seems to be the suitable solution. It is an OMG standard, applied to all the software life cycle, in order to obtain integration and interoperability between different, changing and emergent technologies. MDA represent a new framework to understand, analyze, design, handle and maintain all the aspects of the information systems using open standards and different types of models.

We can find basic foundations of MDA benefits in several books [Hubert, 2002][Kleppe et al., 2003][Raistrick et al., 2004] and works [Harmon, 2002][Dsouza, 2001]. You can also see the MDA web page, www.omg.org/mda, and the success stories in www.omg.org/mda/products success.htm.

The purpose of this paper is to show that, according to the MDA specification, we can begin modeling the system requirements and then, following a well defined method, transform them to get the same model but with the technological details of each platform. The final result will be the equivalent implementation code of all these platforms, obtained in a process that should be automated.
2. GENERAL DESCRIPTION AND KEY POINTS OF MDA

One of the key aspects of MDA is that it is based on the use of proven and successful open standards [Baisley et al., 2000]. Moreover, if we want to obtain good results in software development we need to apply MDA the whole of the software life cycle. In a valid MDA framework [Kleppe et al., 2003] we can manage different types of models, generally based on UML (Unified Modeling Language) profiles, and to which we apply transformations and mapping rules.

2.1 Main MDA models

According to the OMG’s MDA Guide [OMG, 2003] the main MDA models are:

1. **The Computation Independent Model (CIM).** It must be considered a domain model that include the vocabulary of the business. It represents the main system requirements and concepts but without any computational aspect. The CIM focuses on the business, not on the software.

2. **The Platform Independent Model (PIM).** While the CIM focuses on the entire business system, the PIM is only related to the supporting software of this system. In this case, the term platform references the technologies and other irrelevant aspects for the software logic. Therefore the PIM is a formal specification in a certain modeling language, like UML, to show the requirements, structure and functionality of the software but with no definitions related to the platform.

3. **The Platform Specific Model (PSM).** This is the same model than the PIM, because the software logic does not change, but now adding the necessary technological details to implement code in real platforms.

In addition, we will use the term **Final Code Model (FCM)**, to represent the low level model of an information system, that is, the code for a specific platform.

For each PIM can exist several PSM, and for each PSM can exist several FCM, but all of them will have the same business logic. They represent the same information system but from different points of view. This concept is not something new. In theory, according to MDA, we would only have to define a PIM and apply automatic transformations for the platforms that we wish to use. In the future, if we would change our platform, we would only change the parameter of this platform in the transformation process, this is actually something new and it is possible but difficult to reach.

2.2 UML profiles and MDA mappings

The **UML profiles** contain a set of UML extensions which complement the software models in specific domain or platform. This is a basic MDA foundation because UML profiles represent the mechanism to define the valid elements that we can use in particular areas and platforms.

For example, the term protocol is mainly used in telecommunications projects. To model software in this area we should use an UML profile for the telecommunications domain that will contain the definition of the element protocol, not included in the UML language. In addition we can use specific platforms elements, like EntityBean in EJB (Entreprise Java Beans) / J2EE (Java 2 Enterprise Edition) projects. We can also use an UML profile for EJB/J2EE that will contain the definition of this element, obviously not included in UML. Both UML profiles are useful for different definitions.

The UML profile concept is very important in MDA, because represents the way to separate the domain or platform details from the business logic of each problem. In fact, OMG is working hard on it and some of these UML profiles are now standardized. This is not enough because the best situation would be that all the platforms and domains would have their own UML profile standardized.

Profiles define the constructions that we can use in specific domains and platforms. But we need something more because we have to transform and connect the elements from a source model to a target model. We need a **mapping** process between these models, that is, a set of rules, transformation techniques and realizations between elements to apply to source and target models. Therefore several mappings must exist because we have different types of models (PIM-PSM-FCM) and platforms (CORBA, EJB/J2EE, etc.).
3. FROM THE PIM TO THE FCM IN A VALID MDA FRAMEWORK

In this section we expose a very simple example to show how these MDA concepts are necessary to improve integration, productivity and quality. This example supposes that we need to develop a project based on a telecommunications web application. The transformations applied, from the PIM to the FCM according to the MDA specification are exposed in the next points, associated to the schema of figure 1.

1. We must choose a modeling language based on MOF. In this case we select UML to define the domain and vocabulary in the CIM and the software requirements and business logic in the PIM. Normally there is no transformation process or links between CIM and PIM elements and both used to be created manually.

2. We use an UML profile for telecommunications because we suppose this is the application domain of the software that we are developing in the example. The PIM can only include elements based on UML constructions and the profile extensions.

3. The mapping process between PIM and PSM apply a set of mechanisms and rules that define how each UML element must be transformed from the PIM to the PSM for EJB/J2EE platform based on Java components. This is much more than a simple transformation, because every PIM construction must be traced to one or more PSM elements to ensure consistency between models. Additionally, the UML profile must be known for the mapping between PIM and PSM to guide the transformation and validation. Now
this UML profile for EJB/J2EE defines the valid UML extensions for EJB components that we can incorporate in the PSM.

4. The PSM, equivalent to the PIM, will contain EJB components in UML language with details of this platform, according to the constructions, extensions and rules included in the UML profile for EJB/J2EE.

5. The second mapping apply the transformation rules between the PSM and the Java code, called FCM. It also links the PSM elements with the final Java classes. In this case, the transformation should know the UML profile of the platform because this would improve the translation process.

6. The FCM automatically obtained is not the final version of the implementation code. It will need to be completed manually with some parts impossible to generate in an automated process. This FCM represents a part of the application than can be automatically generated giving the previous models.

4. CONCLUSION

MDA guides the software development process using mainly two types of models, PIM and PSM. As we have described, in the final step we can translate the PSM to FCM (code) with development tools. PIM definition simplify the initial high level model because only contains the business logic.

This represent one of the greatest MDA advantages. As the PIM does not have technological details, it will be valid for all platforms. Furthermore, the same model will be always managed, although the future technologies change or disappear. Another MDA advantage resides in how the PIM can be transformed to PSM, and this to the FCM. If we use appropriate UML profiles and good MDA mapping tools, we can improve our software productivity, quality and integration. But all depends on the way MDA tools treat the UML profiles and transformation rules between models. They should be managed as input parameters.

But there are many limitations too. In first place, the PIM should be based and linked to the requirements defined in the CIM. But normally the PIM is created manually. In second place, the PIM must be complete, precise and include semantic information. This is difficult due to the MOF 1.4 and UML 1.5 (included OCL) restrictions, although we think that the new versions, MOF 2.0 and UML 2.0, will reduce these problems. Besides, the number of standardized UML profiles is not enough. In third place, MDA depends much on the tools. Most of the present MDA solutions use their own set of tools, which provides very different MDA interpretations, each one with a special vision of basic concepts like UML profiles and mappings.

MOF 2.0 and UML 2.0 will solve many limitations of previous versions, and this is something very important for the success of MDA. The concepts explained in this document will be the same, but probably will appear more MDA real cases. There are several documents [Hubert, 2002][Kleppe et al., 2003][Frankel, 2003][Harmon, 2002] that show the MDA potential in real problems.

Our future works will study how we can resolve the limitations of the CIM, and how the requirements modeled can be linked to the PIM in an automated process. We think that is one of the most important gaps in MDA and that there is not enough works in this field.

REFERENCES