From Object Composition to Model Transformation with the MDA

Jean Bézivin
University of Nantes
2, rue de la Houssinière, BP 92208
44322 Nantes cedex 3, France

Jean.Bezivin@sciences.univ-nantes.fr

Abstract

The object technology revolution has allowed the replacement of the more than twenty-years old step-wise procedural refinement paradigm by the more fashionable object composition paradigm. Surprisingly this evolution seems itself today to be triggering another even more radical change, towards model transformation. As a concrete trace of this, the Object Management Group (OMG) is rapidly moving from its previous Object Management Architecture vision (OMA) to the newest Model-Driven Architecture (MDA). Some of the main characteristics of this new organization will be outlined in the presentation.

1. Introduction

The OMG has proposed a modeling language called UML (Unified Modeling Language) for describing all kinds of object-oriented software artifacts. The internal architecture and applicability scope of UML are not yet completely stabilized [4]. In order to allow other similar languages to be defined as well, the OMG uses a general framework based on the MOF (Meta-Object Facility [6]). UML and the MOF are the centerpieces of the four-layers modeling stack of the Model-Driven Architecture MDA ([8], [1]). The real status of these modeling layers is still unsettled. One way to look at the architecture is to compare it to the area of programming languages (Figure 1). At the lowest M0 level we find the real world, corresponding to a given execution of say a Pascal program. At level M1 we find the models, corresponding for example to a given Pascal program. For one such Pascal program there is infinity of possible executions. At level M2 we find the meta-models, corresponding for example to a grammar for the Pascal language. For a given grammar, there is infinity of well-formed programs. Finally level M3 is the meta-meta-model level. It may be compared to the self-defined, extended BNF formalism. The EBNF formalism allows defining infinity of grammars. In the standard OMG modeling stack, the MOF at level M3 is self-defined and allows defining meta-models at level M2. The UML meta-model is one of the well-known examples. It allows defining UML models at level M1. A given UML model describes a real phenomenon at level M0, with entities and events unique in time and space.

2. Models Everywhere

The consensus on UML has been instrumental in this transition from code-oriented to model-oriented software production techniques. A key role is now played by the concept of meta-model. The notion of a meta-model is strongly related to the notion of an ontology [2], used in knowledge representation communities. The MOF has emerged from the recognition that UML was one possible meta-model in the software development landscape, but that it was not the only one. Facing the danger of having a variety of different non-compatible meta-models being defined and independently evolving (data warehouse, workflow, software process, etc.), there was an urgent need for a global integration framework for all meta-models in the software development scene. The answer was thus to provide one language for

Published in the proceedings of TOOLS'USA, Volume IEEE TOOLS-39, Santa Barbara, August 2001

defining meta-models, i.e. a meta-meta-model. Each meta-model defines itself a language for describing a specific domain of interest. For example UML describes the artifacts of an object-oriented software system. Some other meta-models may address domains like legacy systems, data warehouses, software process, organization, tests, quality of service, party management, etc. Their number is important and keeps growing, under the control of the end-user and platform working groups. They are defined as separate components and many relationships exist between them.

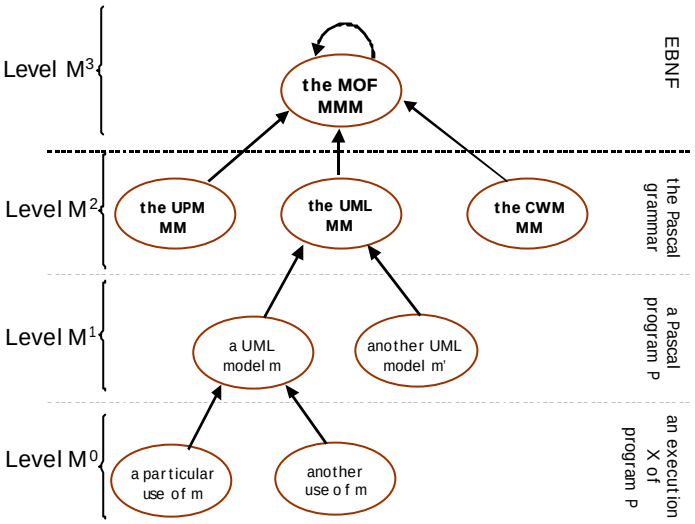

Figure 1 The OMG four layers standard modeling stack

3. Model serialization

The real change in the model engineering happened when it became clear that model could be directly used in software production chains. Although this possibility had often been considered and partially applied, we may now envision its large-scale industrial deployment. Until now object analysis and design models have mainly been used to document software system. Analysts and designers were building models that were provided to programmers only as inspiration material to facilitate the production of concrete software. The move from this "contemplative" period to a new situation where production tools will be model-driven has been facilitated by the introduction of the XMI recommendation [7]. This XMI recommendation builds upon many other standards like UML, MOF, OCL and XML. We may recognize its importance from the fact that many new proposals at OMG are no more provided as a simple paper description, but as a XMI DTD as well, corresponding to the MOF-compatible meta-model of the proposal. This helps to reduce the gap between human readable and computer interpretable standards.

The W3C XML standard provides the transfer syntax but also a complete technological space with widely available and well-engineered tools on which to map the MOF-compatible models. This will allow for example to apply transformation systems like XSLT to any kind of high level models. As Figure 2 suggests, there is a similarity between the relation of a XMI document to a XMI DTD or schema on one side and the relation of a MOF-based model to a MOF-based meta-model on the other side. The XML, MOF, UML and OCL standards are well integrated in XMI and play together to provide a powerful model serialization tool. The move from DTDs to XML schemas is being integrated into this process and will strengthen the resulting possibilities.

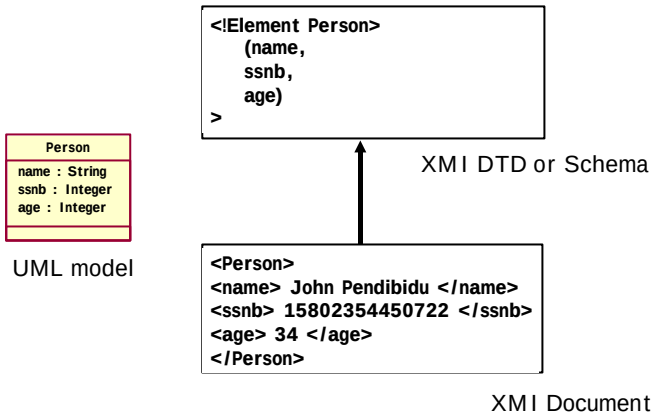

Figure 2 XMI representation of MOF-compatible models

4. Separation of aspects

The MDA is preparing for a new situation where models will be first class entities. They will be stand-alone and on-line accessible (Figure 3). This means that the execution model will contain execution objects and if necessary these execution objects will have the capacity to access other attributes explicitly represented in other models. Ultimately, all entities present in the various models may show autonomous behavior. This organization is based on the fact that there may exist a common execution bus (i.e. CORBA, DotNet, the Web, Java) and an "orthogonal" common representation bus (i.e. the MOF). The architecture suggested by Figure 3 goes much beyond proposals of separation of aspects with AOP (Aspect-Oriented Programming, [3]). It shows how the meta-modeling framework may provide possibilities only currently offered by computational introspection and reflection. Of course the separation of aspects with meta-models will make its way in a progressive evolution. However more limited applications of this general scheme to deal with the reification of contracts, exceptions or performance QoS specification may be envisioned on a medium-term basis.

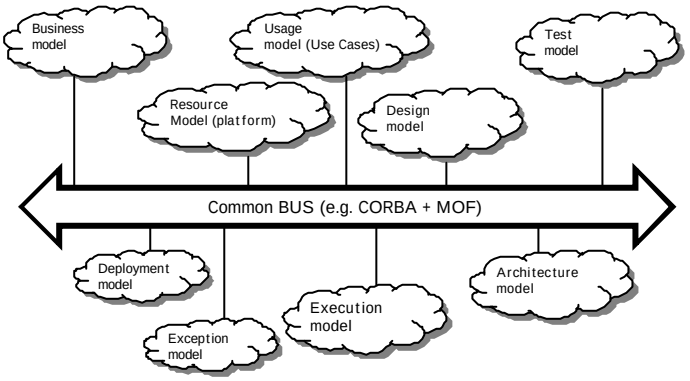

Figure 3 Separation of aspects with a general model infrastructure

5. Middleware generation support

One important functionality is present in the MOF, which is made available to all meta-model branches. This is middleware generation support. Originally this was provided only for the OMG standard middleware, i.e. CORBA. In the new MDA organization, instead of targeting only the CORBA middleware, the integrated facility will allow to generate for a number of different platforms: Sun (Java/EJB), Microsoft (C#/DotNet), the Web, etc. Furthermore any new middleware that may appear in the future can easily be integrated. Let

us take, as an illustrative example, the UML fragment defined in Figure 4. This could produce the corresponding CORBA API for accessing Employee objects in the IDL language. Alternatively we could also generate Java interfaces from the same fragment as illustrated by Figure 4.

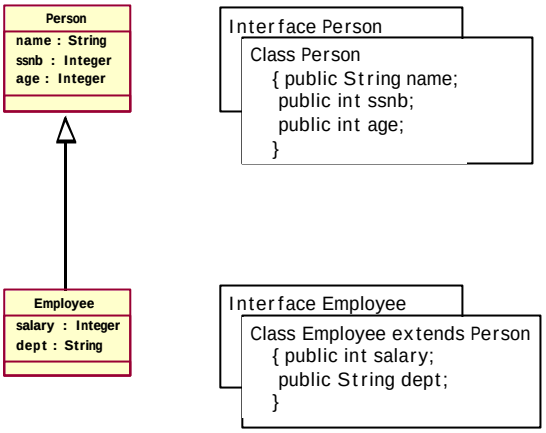

Figure 4 A UML model fragment and a corresponding Java code fragment

Our illustration was a UML fragment. Obviously any model based on a MOF-compliant meta-model would have the same property. This possible generation of IDL APIs from UML models means different things. It will be no more necessary to define end-user recommendations in IDL since the high level UML expression will be able to generate automatically for this target. When we look at the activity spent in this area by various end-user OMG working groups (Transport, HealthCare, Electronic Commerce, etc.), we see the important impact of this move. The fact that the target middleware may be parametrically changed (IDL, Java, C#, the WEB, etc.) offers a lot of way for economy. There are also additional advantages in doing this move since we can inject a variable dose of precision into UML models by adding OCL statements. This is not an all-or-nothing process (i.e. using a formal specification language or not using one at all), but an engineering decision. Of course this was not possible with IDL or with any other common target middleware.

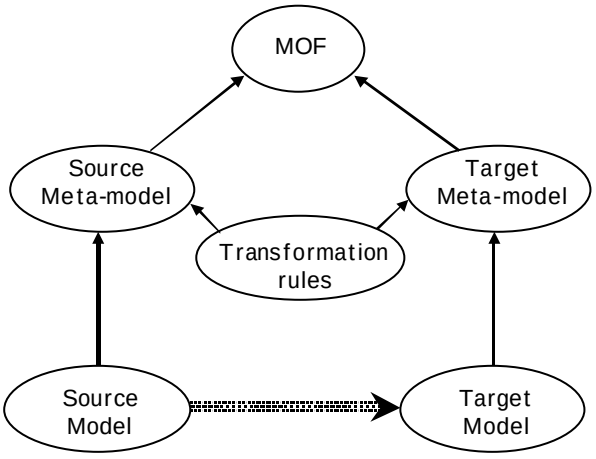

Figure 5 Meta-model based model transformation

6. Model Transformation

The question of model transformation also lies at the center of the MDA approach. The designer and programmer would be given for example the profiles UML for CORBA or

UML for C++ and can then use these dialects of UML to prepare for the transformation between a UML design model and IDL or C++ code, with the help of some limited facilities provided by the UML CASE tool vendors. As a matter of fact the possibilities don't lie there but in more general approaches, as illustrated by Figure 5. A typical proposal has been made in [5]. We may consider that we have here two meta-models. The source one could be UML for example and the target one could be C# ore more realistically a DotNet meta-model. The transformation of the UML model to EJB code may be specified by a set of rules defined in terms of the corresponding meta-models. The expression of these rules may be facilitated if a basic generic framework is present in the MOF. The transformation engine itself may be built on any technology like the XSLT tools.

7. Conclusion

The move from procedural technology to object technology has triggered a more radical change in our way of considering information systems and of conducting software engineering operations. One of the possible evolution paths is called model engineering. It consists in giving a first-class status to models and model elements, similarly to the first class status that was given to objects and classes in the 80s, at the beginning of the object technology era. The essential change is that models are no more used only as mere documentation for programmers, but they can be directly used to drive tools.

OMG was set up twelve years ago to solve the basic problem of object interoperability (how to make heterogeneous software written in C++, Eiffel, Smalltalk, etc. function properly upon various different distributed platforms). The answers have been CORBA, IDL, IIOP, etc. and the OMA distributed programming framework. Today we are facing new and harder interoperability problems and it will probably take a longer time to solve them. What is needed is a sound global model-engineering framework. The MDA organization is an initial answer to this challenge. The OMG four-levels modeling stack is the operational kernel of the MDA.

8. References

[1] Dsouza, D. **Model-Driven Architecture and Integration: Opportunities and Challenges** Version 1.1. February 2001, available at www.kinetiuym.com

[2] Guarino N., Welty, C. **Towards a Methodology for Ontology-based Model Engineering.** in Bézivin, J. and Ernst, J., (eds.), First International Workshop on Model engineering, Nice, France, June 13, 2000, available at www.metamodel.com

[3] Kiczales, G. & al. **Aspect-Oriented Programming** in Aksit, M. and Matsuoka, S. (eds.), 11th European Conference on Object-Oriented Programming, LNCS #1241, pages 220-242, Springer Verlag, 1997

[4] Kobryn, C. **The Road to UML 2.0: Fast track or Detour**. SD Magazine, April 2001.

[5] Lemesle, R. **Transformation Rules Based on Meta-Modeling** EDOC,'98, La Jolla, California, 3-5 November 1998, pp.113-122.

[6] OMG/MOF **Meta Object Facility (MOF) Specification**. OMG Document AD/97-08-14, September 1997. available at www.omg.org

[7] OMG/XMI **XML Model Interchange (XMI)** OMG Document AD/98-10-05, October 1998. available at www.omg.org

[8] Soley, R. and the OMG staff **Model-Driven Architecture**. OMG draft document available at www.omg.org November 2000.