UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org
Vol. V, No. 2, April 2004

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).
UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.
UPGRADE was created in October 2000 by CEPIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

E-mail addresses for editorial correspondence: <rfcalvo@ati.es>, <nicolet@acm.org> or <rcarniel@dgt.uniud.it>

E-mail address for advertising correspondence: <novatica@ati.es>

Upgrade **Newslist** available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Next issue (June 2004):
"Digital Signature"
(The full schedule of **UP**GRADE is available at our website)

UML and Model Engineering

Guest Editors: **Jesús García-Molina, Ana Moreira, and Gustavo Rossi**

Joint issue with NOVÁTICA*

Mosaic

* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by NOVÁTICA, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica/>, and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIS society ALSI and the Italian IT portal Tecnoteca at <http://www.tecnoteca.it>.

In Search of a Basic Principle for Model Driven Engineering

Jean Bézivin

In November 2000, the OMG (Object Management Group) made public the MDA™ (Model Driven Architecture) initiative, a particular variant of a new global trend called model engineering. The basic ideas of model engineering are germane to many other approaches such as generative programming, domain specific languages, model-integrated computing, software factories, etc. MDA may be defined as the realization of model engineering principles around a set of OMG standards like MOF (Meta Object Facility), XMI (XML Metadata Interchange), OCL (Object Constraint Language), UML (Unified Modeling Language), CWM (Common Warehouse Metamodel), SPEM (Software Process Engineering Metamodel), etc. Similarly to the basic principle "Everything is an object" that was important in the 80's to set up the object-oriented technology, we suggest here, in model engineering, that the basic principle "Everything is a model" may be key to identifying the essential characteristics of this new trend.

Keywords: MDA, Meta Model, Model Driven Architecture, Model Driven Engineering.

1 Introduction

An important paradigm shift is happening in the field of software engineering that may have important consequences on the way information systems are built and maintained. As J. Greenfield and K. Short say in [3]: *"The software industry remains reliant on the craftsmanship of skilled individuals engaged in labour intensive manual tasks. However, growing pressure to reduce cost and time to market and to improve software quality may catalyse a transition to more automated methods. We look at how the software industry may be industrialized, and we describe technologies that might be used to support this vision. We suggest that the current software development paradigm, based on object orientation, may have reached the point of exhaustion, and we propose a model for its successor."*

The central idea of object composition is progressively being replaced by the notion of model transformation. One can view these in continuity or in rupture. The idea of software systems being composed of interconnected objects is not in opposition to the idea of the software life cycle being viewed as a chain of model transformations.

2 The OMG Model Driven Architecture

In November 2000, the OMG (Object Management Group) announced its MDA™ (Model Driven Architecture) initiative [8]. The consensus on UML (Unified Modeling Language) 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. But this is not sufficient. The recognition that UML was one possible meta-model in the software development landscape, but not the only one, gave rise to the MOF (Meta Object Facility). To avoid 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 to provide one language for defining meta-models, i.e. a meta-meta-model (level M^3 in Figure 1). Each meta-model defines itself a language for describing a specific domain of interest (level M^2 in Figure 1). 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. Each is important and their numbers keep growing. They are defined as separate components and many relationships exist between them.

The real change in model engineering happened when it became clear that models could be directly used in software production chains. Although this direct use had often been previously considered and partially applied, its large-scale

Jean Bézivin is professor of Computer Science at the University of Nantes, France, member of the newly created ATLAS research group in Nantes (INRIA & CNRS/LINA). He has been very active in Europe in the Object-Oriented community, starting the ECOOP series of conference (with P. Cointe), the TOOLS series of conferences (with B. Meyer), the OCM meetings (with S. Caussarieu and Y. Gallison) and more recently the <<UML>> series of conferences (with P.-A. Muller). He also organized several work-shops at OOPSLA like in 1995 on "Use Case Technology", in 1998 on "Model engineering with CDIF", at ECOOP in 2000 on "Model Engineering", etc. His present research interests include object-oriented analysis and design, product and process modelling, legacy reverse engineering, general model engineering and more especially model-transformation languages and frameworks. <Jean.Bezivin@lina.univ-nantes.fr>

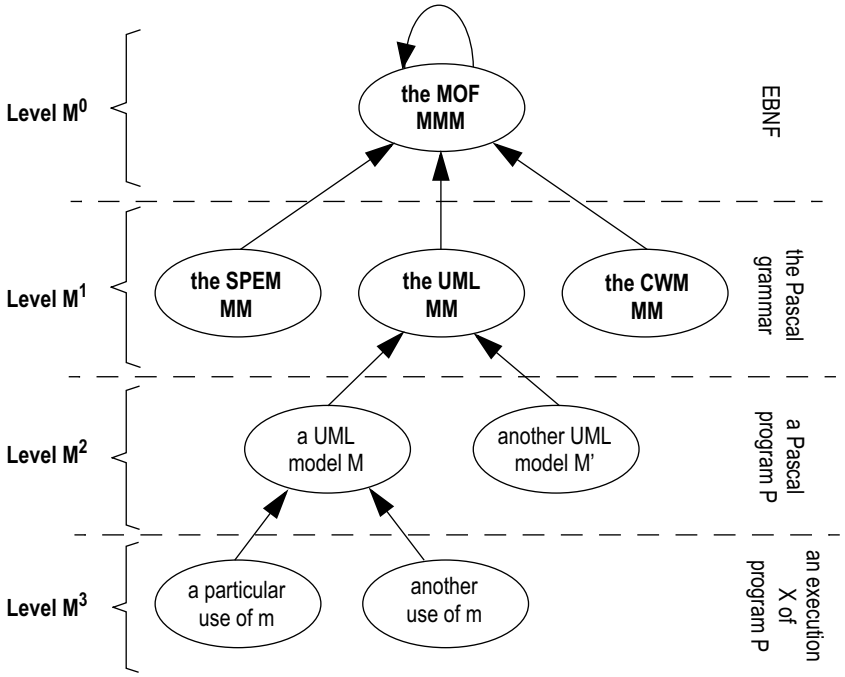

Figure 1: The OMG Four Layers Standard Modeling Stack.

industrial deployment may now be envisioned [3]. Until now object analysis and design models have mainly been used to document software systems. Analysts and designers built 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 standards like the XMI (XML Metadata Interchange) recommendation [7].

The MDA approach is not based on a unique idea. Among the objectives pursued, are the separation from business-neutral descriptions and platform dependent implementations, the expression of specific aspects of a system under development with specialized domain-specific languages, the establishment of precise relations between these different languages in a global framework and, in particular, the capability to express operational transformations between them.

Model transformation is also central to the MDA approach. A proposal was originally made in [4]. A request for proposal [6] is currently undergoing for defining some sort of a "Unified Model Transformation Language". This will allow transforming model *Ma* into another model *Mb*, irrespective of the fact that their corresponding meta-models *MMa* and *MMb* are identical or different. Furthermore, the transformation program, composed of a set of rules as described in Figure 2, should itself be considered as a model *Mt*. As a consequence, it will be based on the meta-model *MMt*, an abstract definition for this unified model transformation language.

These are some elements of the MDA four-level stack. They are rapidly evolving, producing powerful industrial tools that are being applied in particular areas. However, the research community is also trying to understand the concepts and principles underlying this industrial MDA approach. In the next section we look at some of the possible core concepts of model engineering.

3 Basic Principles

A basic principle in object technology ("Everything is an object" [P1]) was most helpful in driving the technology in the

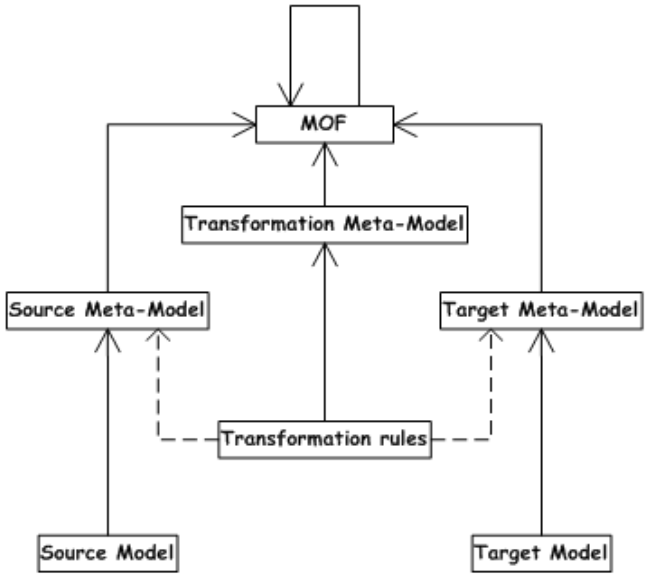

Figure 2: Meta-model Based Model Transformation.

80s, in the direction of simplicity, generality and power of integration. Similarly in model engineering, the basic principle "Everything is a model" [P2] has many interesting properties, among them the capacity to generate a realistic research agenda. We suggest that this may be most useful in understanding many questions about model engineering in general and the MDA™ approach in particular.

Everything is an object	**[P1]**
Everything is a model	**[P2]**

As suggested by Figure 3 and Figure 4, the conceptual tools that were in focus in the 80s are being renewed. In the beginning of object technology, what was important was that an object could be an instance of a class and a class could inherit from another class. This may be seen as a minimal definition in support of principle [P1]. We call the two corresponding basic relations *instanceOf* and *inherits*. Very differently, what now seems to be important, is that a particular view (or aspect) of a system can be captured by a model and that each model is written in the language of its meta-model. This may be seen as a minimal definition in support of principle [P2]. We call the two basic relations *representedBy* and *conformantTo*. It is very likely that the discussions on the exact meaning of these two central relations associated with principle [P2] will take as much time to settle as the sometimes heated initial discussions took, on the two relations associated with principle [P1].

Incorrect use of the old [P1] relations within the new context of model engineering, for example by stating that a model in an *instanceOf* a meta-model, often leads to much confusion and does not help in clarifying a complex evolution.

Recognizing the difference between the two sets of relations [P1] and [P2] will also help to stress that the model engineering and the object technology views of the software development world should not be viewed as opposite but as complementary approaches.

The basic use of a meta-model is that it facilitates the separation of concerns. When dealing with a given system, one may

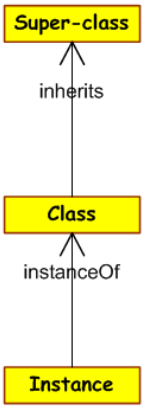

Figure 3: Basic Notions in Object Technology [P1].

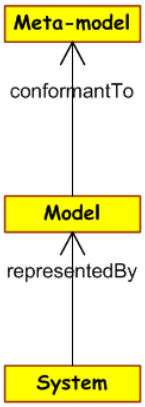

Figure 4: Basic Notions in Model Engineering [P2].

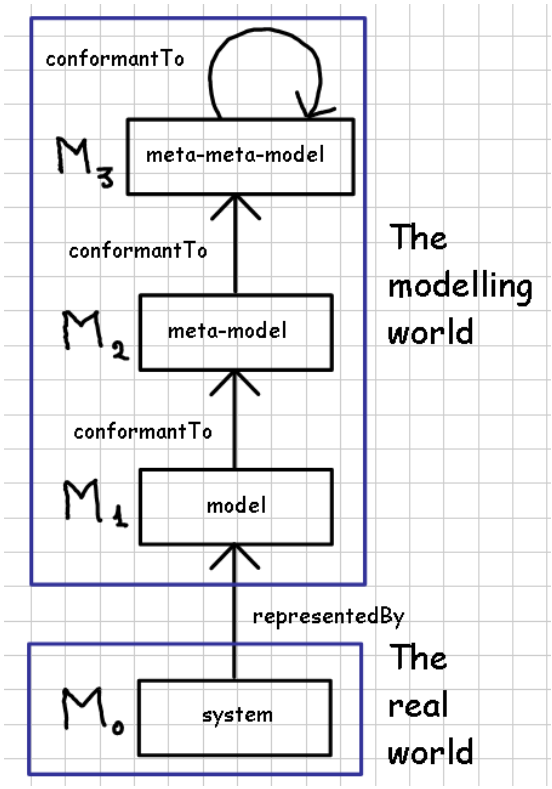

Figure 5: The 3+1 MDA Organisation Revisited.

observe and work with different models of this same system, each one characterized by a given meta-model. When several models have been extracted from the same system with different meta-models, these models remain related and, to some extent, the reverse operation may apply, namely combination of concerns. What we need for that is a clean organization of composite models, in regard to the corresponding composite meta-models.

The organization of the classical four-level architecture of OMG should more precisely be named a 3+1 architecture as illustrated in Figure 5. At the bottom level, the M_0 layer is the real system. A model *represents* this system at level M_1. This model *conforms* to its meta-model defined at level M_2 and the meta-model itself *conforms* to the meta-meta-model at level M_3. The meta-meta-model conforms to itself. This is very similar to the organization of programming languages, as already suggested in the right-hand column of Figure 1. A self-representation of the EBNF (Extended Backus-Naur Form) notations takes some lines. This notation allows defining infinity of well-formed grammars. A given grammar, for example the grammar of the Pascal language, allows defining the infinity of syntactically correct Pascal programs. One Pascal program is a symbolic representation of the infinity of its possible executions. The Pascal program, the Pascal grammar, the EBNF self-described notation are all examples of symbolic models. The execution of the program, on the contrary, is a real phenomenon involving changes of values in electronic memories and much more: it is part of the real world.

The consequence of applying principle [P2] to the software development domain is that we are now observing the appearance of a large number of fine-grained high abstraction meta-models, each one defining a domain specific language. There is a need to organize this huge collection of meta-models, on the conceptual [2] or practical [1] points of views.

The impact on the organization of the software production and maintenance work-bench is also beginning to appear. The UML modelling tool, which was previously at the centre of this workbench, is now only one tool among others, allowing the initial capture of UML models. At the same time, we see a number of new tools appearing, like independent transformation engines and frameworks. All these tools operate on top of a model and meta-model repository. Each of them implements a limited set of specific operations on models and meta-models. Their behaviour is sometimes partially driven by generic uploadable meta-models.

In this rapid evolution of the industrial landscape, we can clearly see the impact of principle [P2] at work. One need not have, for example, only one standard UML to Java transformation option, 'hardwired' in a given UML tool. On the contrary, one will use this UML tool to capture UML models, and then, from a large library of transformations, choose the ones that would suit its own goal and environment. If none is found, one may specialize some of them to correspond to its particular needs. Besides transformation tools, there will also be verification, metrication, legacy recovery, model weaving tools and many more. All these tools will uniformly access models, meta-models, and their elements through a common standard repository API. This regular industrial organization of the tool cooperation, in the software production chain, is made possible because of the unification power of models.

4 Conclusions

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 evolutionary paths is called model engineering. This consists in giving 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 longer used only as mere documentation for programmers, but can now be directly used to drive software production tools.

Considering everything as a model in a software development approach has many consequences that are progressively emerging. The MDA is one of the first industrial large-scale applications of this principle. High synergy between these industrial application experiments and general research consideration in the field is likely to produce long term improvement in the domain of software production methods.

Acknowledgements

I would like to thank the members of the CNRS MDE/AS group for participating in various discussions on the topics. The responsibility for how their suggestions were interpreted (or misinterpreted) in this version remains, of course, entirely mine. I also want to specially thank Jean-Marc Jézéquel, Frédéric Jouault, Joaquin Miller and Bernard Rumpe for many clarifying discussions on these subjects.

References

[1] J. Bézivin, S. Gérard, P. A. Muller, L. Rioux. MDA Components: Challenges and Opportunities. Metamodelling for MDA Workshop, York, 2003, <http://www.sciences.univ-nantes.fr/info/lrsg/Pages_perso/JB/Jean.Bezivin.html>.

[2] J. Bézivin, O. Gerbé. Towards a Precise Definition of the OMG/MDA Framework. ASE'01, Automated Software Engineering, San Diego, USA, November 26–29, 2001, <http://www.sciences.univ-nantes.fr/info/perso/permanents/atl/publications/ ASE01.OG.JB.pdf>.

[3] J. Greenfield, K. Short. Software factories Assembling Applications with Patterns, Models, Frameworks and Tools. OOPSLA'03, Anaheim, Ca., USA.

[4] R. Lemesle. Transformation rules based on meta-modelling. EDOC'98, San Diego, 3–5 November 1998, <http://www.sciences.univ-nantes.fr/info/lrsg/Pages_perso/RL/Publications/EDOC98-lemesle.pdf>.

[5] Object Management Group: OMG/MOF Meta Object Facility (MOF) Specification. September 1997, <http://www.omg.org/docs/ad/97-08-14.pdf>.

[6] Object Management Group: OMG/RFP/QVT MOF 2.0 Query/Views/Transformations RFP. October 2002, <http://www.omg.org/docs/ad/02-04-10.pdf>.

[7] Object Management Group: XML Model Interchange (XMI). October 1998, <http://www.omg.org/docs/ad/98-10-05.pdf>.

[8] R. Soley and the OMG staff. Model Driven Architecture. November 2000, <ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf>.

