ISSN 0832-7203

TOWARDS A PRECISE DEFINITION OF THE OMG/MDA FRAMEWORK.

Par : Jean Bézivin, Nantes, France
Olivier Gerbé, HEC Montréal

Cahier du GReSI no 01-10

Octobre 2001

Towards a Precise Definition of the OMG/MDA Framework.

RÉSUMÉ

Il y a actuellement un changement de paradigme dans le domaine du développement de systèmes d'information. Plus précisément, nous passons de la technologie des objets et composants vers celle des modèles. La révolution objet a permis de remplacer le vieux paradigme de décomposition procédurale par celui de la composition d'objets. Cette évolution a engendré une autre révolution peut être plus radicale vers la transformation de modèles. Comme preuve de cette nouvelle révolution, l'Object Management Group est passé d'une vision centrée sur les objets (Object Management Architecture) à une vision centrée sur les modèles (Model-Driven Architecture). Cet article présente une interprétation de cette évolution. Pour rester aussi indépendant que possible du langage utilisé pour décrire ce nouveau paradigme, nous avons choisi le formalisme neutre des graphes conceptuels développé par John Sowa. Ceci nous a permis de voir des problèmes potentiels et de suggérer quelques pistes de solutions.

MOTS-CLÉS

Modèles; Meta-modèles; Graphes conceptuels; UML; MOF; MDA

ABSTRACT

We are presently witnessing an important paradigm shift occurring in the area of information system construction, namely from object and component technology to model technology. The object technology revolution has allowed the replacement of the more than twenty-years old step-wise procedural decomposition paradigm by the more fashionable object composition paradigm. Surprisingly this evolution seems itself to have triggered today another even more radical change, towards model transformation. As a concrete trace of this, the Object Management Group is rapidly moving from its previous Object Management Architecture vision to the newest Model-Driven Architecture. This paper presents an abstract investigation into the interpretation of this evolution. In order to stay as language independent as possible, we have decided to use the neutral formalism of Sowa's conceptual graphs to describe the various situations that characterize this organization. This will allow us to identify some potential problems in the proposed modeling framework and to suggest some possible solutions.

KEYWORDS

Models; Meta-models; Conceptual Graphs; UML; MOF; MDA

Towards a Precise Definition of the OMG/MDA Framework

Jean Bézivin
LRSG, Université de Nantes
2, rue de la Houssinière, BP 92208
44322 Nantes cedex3, France
Jean.Bezivin@sciences.univ-nantes.fr

Olivier Gerbé
HEC - Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7
Olivier.Gerbe@hec.ca

Abstract

We are presently witnessing an important paradigm shift occurring in the area of information system construction, namely from object and component technology to model technology. The object technology revolution has allowed the replacement of the more than twenty-years old step-wise procedural decomposition paradigm by the more fashionable object composition paradigm. Surprisingly this evolution seems itself to have triggered today another even more radical change, towards model transformation. As a concrete trace of this, the Object Management Group is rapidly moving from its previous Object Management Architecture vision to the newest Model-Driven Architecture. This paper presents an abstract investigation into the interpretation of this evolution. In order to stay as language independent as possible, we have decided to use the neutral formalism of Sowa's conceptual graphs to describe the various situations that characterize this organization. This will allow us to identify some potential problems in the proposed modeling framework and to suggest some possible solutions.

1 Introduction

Our purpose here is to understand the extent and the real meaning of the recent move from object-based to model-based architectures of information system. Our starting point will be the study of a proposed new vision of the Object Management Group (OMG) called Model Driven Architecture (MDA) [14, 8]. The OMG has proposed a modeling language called UML (Unified Modeling Language [13]) for describing all kinds of object-oriented software artifacts. The applicability scope of UML is not yet completely stabilized. In order to allow other similar languages to be defined as well, the OMG is using a general framework based on the MOF (Meta-Object Facility [12]). Some confusion may exist in the application of these conceptual tools may be because they are self-defined and mutually dependent, as we shall see later. In order to well understand them, we will use in this paper an external and neutral formalism to describe the situation and to identify some problems that may arise. The formalism will be Sowa's conceptual graphs [20, 21], which has the property to be simple and precise.

The paper focuses on the peculiarities of new model-centered frameworks. In section 2 we present the formalism of conceptual graphs. We describe in section 3 the main characteristics of the OMG/MDA framework, which is succeeding to the OMA (Object Management Architecture). Section 4 introduces the framework we will use in Section 5 to discuss a number of open questions in applied model engineering and to try to put forward some answers. In the conclusion we summarize the original contribution of this work.

2 Presentation of Conceptual Graphs

This section presents the formalism of conceptual graphs. We provide here only a minimal introduction in order to understand the rest of the paper. More details on conceptual graphs can be found in [21, 5] or in the various annual ICCS conferences on the subject.

J. Sowa introduced conceptual graphs (CGs) in 1984. They are based on a graphical representation system for logic (existential graphs) invented at the beginning of the 20th century by C. S. Pierce [16]. The fact that the company BricABrac employs the employee John Pendibidu may be described in CGs by the following linear textual description:

[Company : BricABrac]→ (emp)→ [Employee : John Pendibidu]

This should be considered as a shorthand for the graphical representation of Figure 1.

The fact that some company employs some employee may be described by:

[Company:*x]→ (emp)→ [Employee:*y]

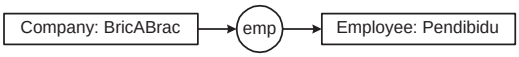

Figure 1. Graphical Representation of a conceptual graph.

There exists an operator called ϕ that allows translating from CGs to first order predicate logic. The previous statement would be rendered as:

$$(\exists x)(\exists y)((Company(x) \wedge Employee(y) \wedge employs(x,y))$$

where the binary predicate employs corresponds to the relation emp of the previous descriptions. Other work establishing the foundations for the correspondence between CGs and First Order Predicate Logic may be found in [1].

A conceptual graph is composed of concepts and relations. Concept nodes are organized as a lattice with a corresponding ordering relation. A concept type corresponds to a definition graph to which an instance of the concept should conform.

Following is the definition of the concept type Employee.

Type Employee(x) is
[Person:*x]← (emp)← [Company:*y]

This means that an employee is a person working for a company.

We assume that CGs are the global modeling language used in the present work to describe the various recommendations of OMG (Figure 2) like UML, MOF, SPE (Software Process engineering), CWM (Common Warehouse Metadata), Wfl (Workflow), etc. This formalism may be over dimensioned for the task at hand, but we see this more as an advantage than a drawback. In order to describe a situation where standards such as UML or the MOF define themselves as competing modeling languages as well, it is wise to base our work on an independent formalism that has at least the same expressive power and preferably more, which is the case for CGs.

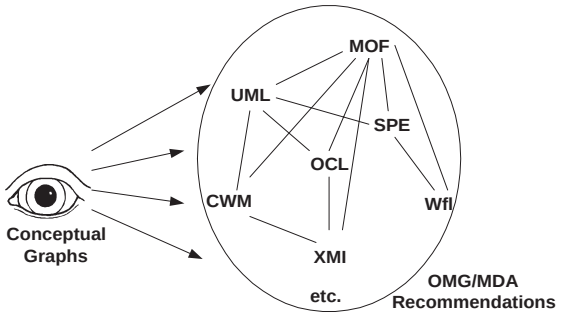

Figure 2. MDA observation with CGs.

3 From OMA to MDA

3.1 Systems, models and meta-models

The most important word in Model Driven Architecture is model and we first need to give a definition of what a model is. A model is a simplification of a system built with an intended goal in mind (Figure 3). The model should be able to answer questions in place of the actual system. The answers given by the model should be the same as the answers that would be given by the system itself, with the condition that questions are in the domain defined by the general goal of the system.

In order to be useful, a model should be easier to handle than the original system. To achieve this, many details from the source system are abstracted out and only a few are selected into the target model. This simplification (or abstraction) is the essence of modeling. Modeling is one of the most common human activities as it usually precedes action. Now if one needs exactly the same modeling operation to be applied several times or by different people, then one should use a meta-model.

A meta-model is an explicit specification of an abstraction (simplification). It uses a specific language for expressing this abstraction: CGs, KIF[10] or MOF are possible candidate languages for this task. In order to define the abstraction, the meta-model identifies the list of relevant concepts and the list of relevant relationships between these concepts. This is what we called the terminology in Figure 3. In some cases it may be sufficient, but in many situations it needs to be completed by a set of logical assertions. Some languages like KIF have the property of being able to express both the terminology and the assertion layers. In other cases like with MOF, we need to add a specific formalism for the assertions (OCL). Figure 3 illustrates relationships between systems, models and meta-models.

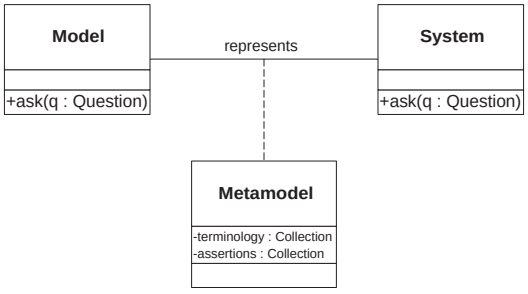

Figure 3. Relations between a system and a model.

A meta-model defines a set of concepts and relations between these concepts and is used as an abstraction filter in

a particular modeling activity. The notion of meta-model is strongly related to the notion of ontology [15], used in knowledge representation communities. From a system, we can extract a particular model with the help of a specific meta-model or ontology.

When dealing with a system, we can observe and work with different models of the same system, each one represented using a different meta-model. Obviously when several models have been extracted from the same system using different meta-models, these systems remain related. Other operations on models are possible, mainly those based on transformation operations. In order to transform models we need a regular organization of models and composite models (models composed of several models), including corresponding meta-models.

Before proposing such an organization, we need to get a deeper understanding of the relation between a model and its meta-model. More precisely, the question addressed here is about the nature of the information contained in a meta-model and about the role played by a meta-model.

3.2 Meta-modeling layers

After the definition of UML, we may now observe a new wave of proposals at OMG, characteristics of a new period and of a new vision. At the center of this evolution, there is the MOF, unique and self-defined meta-meta-model. The concept of a MOF has progressively emerged in the last ten years from the work of different communities like CDIF [6, 9] and IRDS [17], as a framework to define and use meta-models [11, 7]. The MOF came out from the fact that UML was one meta-model in the software development landscape, but 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 a language to define meta-models, i.e. a meta-meta-model. This is the purpose of the MOF. As a consequence, a layered architecture has been defined.

This layered architecture has the following levels.

- M3: the meta-meta-model level (contains only the MOF)
- M2: the meta-model level (contains any kind of meta-model, including the UML meta-model)
- M1: the model level (any model with a corresponding meta-model from M2).
- M0: the concrete level (any real situation, unique in space and time, represented by a given model from M1).

A parallel may be drawn with the domain of formal programming languages (see the right part of Figure 4). Level M3 corresponds to the meta-grammar (like the EBNF notation for example). Level M2 corresponds to the grammar level. Level M1 correspond to the program level. Level M0 corresponds to one given dynamic execution of a program. Level M0 does not correspond to the modeling world (it does not contain model elements but real or imaginary situation items and facts). A given execution of a program at level M0 is not a model by itself; it is depicted by a model (the source code of the program that describes the infinity of possible different executions of this program). Exactly the same situation holds for the four OMG meta-modeling layers.

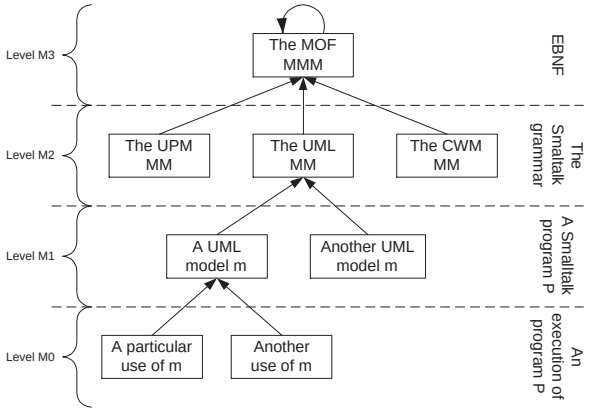

Figure 4. Several spaces, pertaining to different levels.

4 A framework for comprehension

We present in this section the framework we will use to illustrate our unserstanding of MDA. We will utilize the conceptual graph formalism introduced in Section 2 to define our framework.

Figure 5 presents our framework. It is based on the MOF and we have renamed concepts to make the description clearer and to avoid using the same vocabulary to describe the vocabulary itself. We will call NODE the MOF::Class and LINK the MOF::Association. These notions used in the MOF, at level M3, apply to every part located at level M3 and below. The MOF::Specialize relationship is represented by our super relation. Although not directly present in the MOF, we have also made explicit an instantiation relation we call meta in order to clarify the situation.

Using the conceptual graph linear form, the framework of Figure 5 may be written as follows :

[NODE:NODE]→ (meta)→ [NODE:NODE]

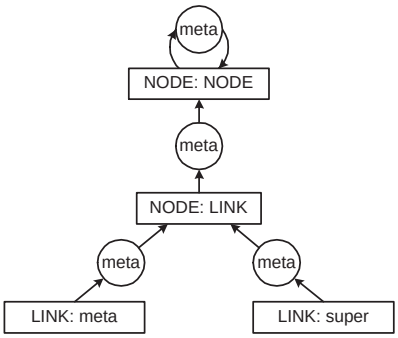

Figure 5. Our framework.

[NODE:LINK]→ (meta)→ [NODE:NODE]
[LINK:meta]→ (meta)→ [NODE:LINK]
[LINK:super]→ (meta)→ [NODE:LINK]

The concept NODE is an ”instance” of itself. The concept LINK is an ”instance” of NODE, concepts meta and super are ”instances” of LINK. We can see that the notation of concepts in CGs is a shortcut to state that an element is an ”instance” of an other element as in [LINK:super] that states that super is an instance of LINK.

Our framework is taking shape, but is still incomplete. One particular missing element is what we will call a CONTEXT which corresponds to a MOF::Package and has some similarity with a CG context. The three contexts: MODEL, METAMODEL and METAMETAMODEL underlying our framework have not been represented in the drawing. They are however real entities.

[NODE:CONTEXT]→ (meta)→ [NODE:NODE]
[NODE:MODEL]→ (meta)→ [NODE:CONTEXT]
[NODE:METAMODEL]→ (meta)→ [NODE:CONTEXT]
[NODE:METAMETAMODEL]→ (meta)→ [NODE:CONTEXT]

In order to have a better understanding of the use of the framework, we give in Figure 6 a complete picture from level M3 to level M0. We state that, in a given model (at level M1), which corresponds to a particular Smalltalk program, the Smalltalk object Mary is an instance (in the sense of the Smalltalk language) of the Smalltalk class Person. In the upper layer (at level M2), we find the elements of the Smalltalk meta-model, namely the concept of Instance, of Class and the relation instance of (StkInstOF) between Class and Instance.

This example will allow answering many questions about the organization of models and meta-models as we will see in the next section.

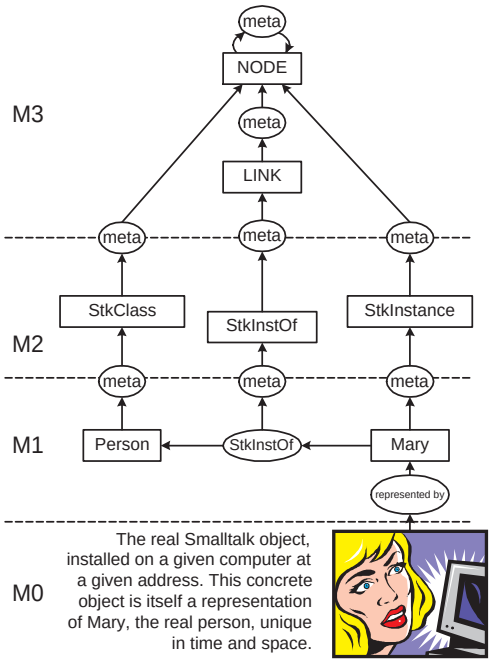

Figure 6. A complete picture.

5 Some Central Issues in Model Engineering

Now that the OMG scene is at least partially described, we can start discussing its strong points and potential remaining problems. There is an important literature about modeling layers: how many layers do we need? May we have more than four layers? Is there a fundamental difference between a model and a meta-model? Is there a fundamental difference between a meta-model and a meta-meta-model? Can a model specialize another model? Can a meta-model specialize another meta-model? etc. We don’t intend to answer here all the questions, but only to highlight some part of the overall model organization problem.

We have seen that metamodels are situated at level M2. In order to avoid possible confusion and to be more illustrative, we will not take the classical example of the UML meta-model, but a Smalltalk metamodel. We could just as well have taken Java or C# to illustrate our presentation.

5.1 The double instantiation problem

One of the hottest discussions is about the double instantiation problem [3]. Is it possible for an entity to be, at the same time, an instance of several classes? In other words in Figure 8, to the question ”what is Mary” should we answer ”Mary is a Person” or ”Mary is a Smalltalk instance” or both? A simple look at Figure 8 is sufficient to understand that this is a non-problem.

This situation could be described as follows in the CG

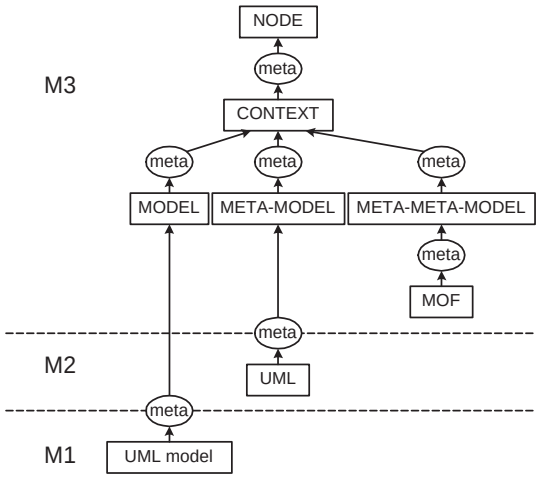

Figure 7. MOF and UML represented in our framework.

linear form:

[NODE:StkInstance]→ (meta)→ [NODE:NODE]
[NODE:StkClass]→ (meta)→ [NODE:NODE]
[NODE:StkInstOf]→ (meta)→ [NODE:LINK]
[NODE:Person]→ (meta)→ [NODE:StkClass]
[LINK:StkInstOf]→ (meta)→ [NODE:StkInstOf]
[NODE:Mary]→ (meta)→ [NODE:StkInstance]
[NODE:Mary]→ (StkInstOf)→ [NODE:Person]

Mary has two definitions here: a local (contextual) one and a global one. The global definition ("Mary is a Smalltalk instance") uses the underlying global typing system of the MOF whereas the local definition ("Mary is a Person") uses the context of the defining Smalltalk meta-model that we are using. There is no ambiguity. If we were using the UML meta-model instead of the Smalltalk meta-model, Mary could be a Person in this context, Person being a UML class.

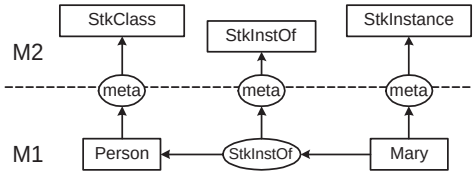

Figure 8. The difference between contextual and global definition.

We clearly see that we can have many local instanceOf relations. These should not be confused with the unique and global type/instance relation that we have named meta here and that corresponds to the MOF typing hierarchy. In Figure 6 we have presented a fragment of a Smalltalk model (i.e. program) at level M1. This model is constrained by the Smalltalk meta-model at level M2. The situation is obviously simplified. In order to pursue the illustration, we could have noticed that the Smalltalk language allows dealing explicitly with meta-classes. We could have added the following element:

[NODE:StkMetaClass]→ (meta)→ [NODE:NODE]
[NODE:Person]→ (StkInstOF)→ [NODE:Person class]

Notice that this new statement in the meta-model requires the notion of meta-class (StkMetaClass) to be defined in the Smalltalk meta-model. The statement above shows however that the relation between a class and its meta-class is identical to the relation between an instance and its class (StkInstOF).

Of paramount importance is the fact that this relation StkInstOf does not cross a hypothetical boundary between layer M1 and M0. If this was not the case, we can see that the result would be an arbitrary number of levels as the relation between the Smalltalk class Person and the Smalltalk meta-class Person class would also cross another boundary between meta-modeling layers.

5.2 Explicit specification

Again here our illustration of Figure 6 is very limited. We could have included also another Smalltalk class named AnimatedBeing and made Person inherit of AnimatedBeing in the Smalltalk sense.

[NODE:Person]→ (StkInherits)→ [NODE:AnimatedBeing]
[NODE:AnimatedBeing]→
(StkInstOf)→ [NODE:AnimatedBeing class]

The point here is that the relation StkInherits represents the inheritance relation of the Smalltalk language. There are plenty of such relations in various environments and they are all different –similar but different–. If we were in a Java environment, we would call this relation JavaExtends and show how it would apply (differently) to the notion of JavaClass and JavaInterfaces. There are plenty of similar but different local inheritance (or specialization or generalization, or extension) relationships. These should not be confused. Moreover they should not be confused with a global specialization relation similar to the MOF Specialize relation defined at level M3 that we have named here super. Making all these relations different improves the precision of the various models. This seems to be a necessary policy if we want to avoid confusion in the discussion.

As we have seen, there is a lot to be gained from a precise and explicit definition of a meta-model[4]. For example, the fact that for any Smalltalk class "the super-class of its meta-class is the meta-class of its super-class" could be written in CGs. Such an assertion could also be added to the Smalltalk meta-model in OCL.

5.3 Relationships between a model and a meta-model

Models and meta-models are different kinds of contexts. They delimit local spaces in the global knowledge context. The basic property of these spaces is that there is no overlapping allowed. Embedding is however possible since a context may contain any element, including another context.

Figure 9 illustrates relationships between a model X and its meta-model Y. Let us consider model X containing entities a and b. There exists one (and only one) meta-model Y defining the "semantics" of X. The relationship between a model and its meta-model (or between a meta-model and its meta-meta-model) is called the sem relationship.

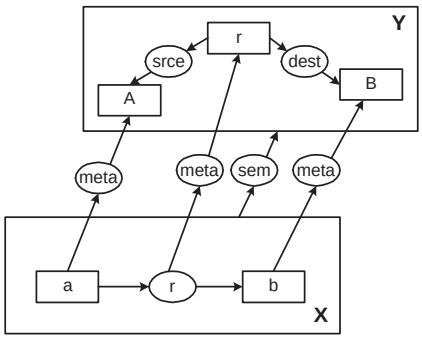

Figure 9. Basic relations between a space X and a meta-space Y.

The meaning of the sem relationship is the following. All entities of model X find their definition in meta-model Y. Relationships meta and sem are mutually related. If an entity of a model X has a meta relationship with an entity of a meta-model Y then X and Y are linked by a sem relationship.

The situation illustrated in Figure 9 may be stated by the following assertions in CGs.

[NODE:X]→ (meta)→ [NODE:MODEL]
[NODE:Y]→ (meta)→ [NODE:METAMODEL]
[NODE:X]→ (sem)→ [NODE:Y]
[NODE:r]→ (srce)→ [A]
[NODE:r]→ (dest)→ [NODE:B]
[LINK:r]→ (meta)→ [NODE:r]
[NODE:a]→ (meta)→ [NODE:A]
[NODE:b]→ (meta)→ [NODE:B]
[NODE:a]→ (r)→ [NODE:b]

The meta relation may be considered as global and basic. The sem relation is derived from the meta relation. The sem relation allows to know at which level a space stands. if it is at level M1, it is a model, at level M2 it is a meta-model and at level M3 it is a meta-meta-model. Relationships meta and sem are different. The approach known as loose meta-modeling consider that these relations are identical. The loose approach brings a lot of problems and is more and more often replaced by the strict meta-modeling approach [2].

It is clear that there are many reasons to avoid stating that: "a model is an instance of a meta-model because its elements are instances of the meta-model elements". It is also clear that the following assertion is wrong:

[MODEL:X]→ (meta)→ [METAMODEL:Y]

because it was previously stated that:

[MODEL:X]→ (meta)→ [NODE:MODEL]

The relationship meta is unique in the sense that a node may have one and only one meta-node in the global context as seen in Section 5.1

5.4 What is a layer?

The global modeling context is thus composed of three kind of local spaces : model, meta-model and meta-meta-model.

About meta-modeling layers, there are many ways to look at the problem. Some of them have already been introduced

- There is a long tradition of using a layered architecture in information systems (e.g. IRDS) and CASE tools engineering (e.g. CDIF). Multiple independent efforts have converged towards similar architectures.
- It is possible to establish an analogy with other domains like the one mentioned of formal grammars for programming languages.
- The decomposition is a natural one, i.e. layer M3 is universal for the entire information systems/software engineering field. Every feature that could potentially be useful to all meta-models should be put at level M3. There is no absolute way to distinguish between a model and a meta-model. This is a problem of point of view, corresponding to the precise task performed. This should be taken as an additional argument to stick to a clear separation between layers M1, M2 and M3.

In addition to all these arguments, it is possible to establish a formal characterization of the notion of meta-modeling layer. The basis is the sem relation presented in Figure 9. The contexts related by the semrelation form a hierarchy. The one at the top is the only self-defined one:

[UNIVERSE]→ (sem)→ [UNIVERSE]

The context UNIVERSE is the MOF in our MDA organization. From there we can deduce which contexts lie at level M2 and which lie at level M1 from their distance to the MOF measured with the sem relation. The situation is a bit more complex that has been described above, because we need to take into account variants of meta-models as discussed below. It is however possible to establish a formal characterization of the sem relation in terms of the meta relation and of the appartenance of a context to one of the three layers. This study has been made in [19] with the demonstration of the three layers conjecture.

5.5 What is a transformation?

The question of model transformation lies at the center of the MDA approach. The above discussion on profiles may convey the idea that this remains very informal. 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.

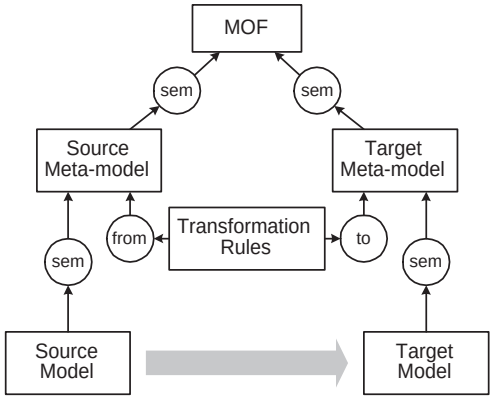

Figure 10. meta-model based model transformation.

As a matter of fact the possibilities do not lie there but in more general approaches that are being studied in many places. A typical proposal has been made in [18] for example. We may consider that we have here two meta-models. The source one could be UML for example and the target one could be Java ore more realistically the EJB meta-model. The transformation from a 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. Suggestions for building this may be found in the CWM meta-model. The transformation engine itself may be built on any technology like the XSLT tools.

Of course, the immediate question that arises when studying the diagram of Figure 10 is about the status of the Transformation Rules Context. Does it have the status of a meta-model as well? Should it use basic facilities (transformation primitives) provided by the MOF? These are typical questions that remain open in the definition of the MDA framework.

6 Conclusion

Within many environments like the OMG, meta-model technologies are now becoming ready for prime time. 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 80's, at the beginning of the object technology era. Two important areas of interest are models of software components and models of software processes. The move from implicit to explicit is characteristic of model engineering. 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. This is the central motto of the new MDA organization proposed by the OMG. A consequence is that it will be possible to separate more clearly the various business models and the various technical models (platform).

The correspondence between meta-models and formal grammars could and should be pursued much further than it has been done here. We clearly see that there are a lot of similarities. The notions of a terminal and a non terminal exist in the formal grammar of the Java language as well as in the UML meta-model.

We have shown in this paper that the notion of modeling layer is quite different from the notion of abstraction layer. The evidence for this is based on the precise kind of relation between two adjacent layers. We have proposed a strict interpretation of these layers where there are exactly three modeling layers. Our interpretation is based on the fact that the so-called fourth layer is not a model, but the system itself (the situation being modeled). As a consequence, the relation between an element of a system at level M0 and an element at level M1 may be given any name like representedBy, but certainly not instanceOf.

Our interpretation of the layered meta-modeling architecture does not correspond to the conventional view but does not contradict it. It has many advantages and one of them being to allow closing the debate about the variable number of meta-modeling layers. As is clearly shown in Figure 8, a new layer is not created whenever a relation instanceOf is found.

Another contribution of this work has been to make a clear distinction between a unique global typing relation called here meta and defined at level M3 and a lot of other contextual relations often called instanceOf and usually defined inside various meta-models. The confusion between all these typing relations has also created many doubts about the organization of meta-modeling layers.

Among the problems that remain to be solved we may mention a precise definition of a meta-model profile, of a model profile and even of a meta-meta-model profile.

7 Acknowledgements

Many issues presented here have benefited from discussions and common work with Richard Lemesle and Erwan Breton. Mikael Peltier has provided much insight on the implementation of meta-model driven model transformation. Valuable suggestions for improving this paper have been made by Mariano Belaunde, Guy Genilloud, Joaquin Miller and Thomas Kuehne,

References

[1] G. Amati and I. Ounis. Conceptual Graphs and First Order Logic. *The Computer Journal*, 43(1), 2000.

[2] C. Atkinson. Supporting and Applying the UML Conceptual Framework. In J. Bézivin and P. A. Muller, editors, *Proceedings of UML'98, Beyond the Notation*, Mulhouse, France, 1998. Springer Verlag. LNCS 1618.

[3] C. Atkinson and T. Kühne. The Essence of Multilevel Metamodeling. In *Proceedings of UML'2001 Conference on Modeling Languages, Concepts and Tools*, Toronto, Ontario, Canada, October 1-5 2001.

[4] M. Belaunde. A Standalone Metamodel for Expressing Model Relationships. 2001. Available at http://universalis.elibel.tm.fr/publications/ RelationshipMetamodelV02.pdf.

[5] M. Chein and M.L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'intelligence artificielle*, 6(4):365–406, 1992.

[6] CDIF Technical Committee. *CDIF: Case Data Interchange Format, Framework for Modeling and Extensibility*. Electronics Industry Associate, 1994. Interim Standard EIA/IS-107.

[7] S. Crawley, S. Davis, J. Indulska, S. McBride, and K. Raymond. Meta-Meta is Better-Better. October 1997.

[8] D. DSouza. *OMG's MDA, An Architecture for Modeling*, 2001. OMG document available at http://www.omg.org/mda/presentations.htm.

[9] J. Ernst. *Introduction to CDIF*, 1997. Available at www.metamodel.com.

[10] M. Genesereth. *Knowledge Interchange Format - draft proposed American National Standard (dpANS) NCTIS.T2/98-004*, 1998. Available at http://logic.stanford.edu/kif/kif.html.

[11] O. Gerbé and B. Kerhervé. Modeling and Metamodeling Requirements for Knowledge Management. In J. Bézivin, J. Ernst, and W. Pidcock, editors, *Proceedings of OOPSLA Workshop on Model Engineering with CDIF*, Vancouver, Canada, October 1998.

[12] Object Management Group. *Meta Object Facility (MOF) Specification*, 1997. OMG Document AD/97-08-14.

[13] Object Management Group. *Unified Modeling Language Specification*, 1999. OMG Document AD/99-06-08.

[14] Object Management Group and R. Soley. *Model-Driven Architecture*, 2000. OMG document available at www.omg.org.

[15] N. Guarino and C. Welty. Towards a methodology for ontology based model engineering. In *International Workshop on Model Engineering (in conjunction with ECOOP'2000)*, Nice / Sophia Antipolis, France, June 2000. Available at http://www.metamodel.com/IWME00/program.html.

[16] N. Houser, D. Roberts, and J. Van Evra. *Studies in the Logic of Charles Sanders Peirce*. Indiana University Press, 1997.

[17] ANSI IRDS. *Conceptual Schema and Modeling Language Analysis*, 1993. Technical Report X3H4/93-196.

[18] R. Lemesle. Transformation Rules Based on Metamodeling. In *Proceedings of EDOC'98*, pages 113–122, La Jolla, CA, November 1998.

[19] R. Lemesle. *Techniques de modélisation et de métamodélisation*. PhD thesis, Université de Nantes, 2000.

[20] J. Sowa. *Conceptual Structure: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[21] J. Sowa. *Knowledge Representation : Logical, Philosophical, and Computational Foundations.* BrooksCole, 2000.