

# Foundations of Model (Driven) (Reverse) Engineering : Models

*Episode I: Stories of The Fidus Papyrus and of The Solarus<sup>1</sup>*

Jean-Marie Favre

ADELE Team, Laboratoire LSR-IMAG  
Université Joseph Fourier, Grenoble, France<sup>2</sup>

**Abstract.** Model Driven Engineering (MDE) received a lot of attention in the last years, both from academia and industry. However, there is still a debate on which basic concepts form the foundation of MDE. The Model Driven Architecture (MDA) from the OMG does not provided clear answers to this question. This standard instead provides a complex set of interdependent technologies. This paper is the first of a series aiming at defining the foundations of MDE independently from a particular technology. A megamodel is introduced in this paper and incrementally refined in further papers from the series. This paper is devoted to a single concept, the concept of model, and to a single relation, the RepresentationOf relation. The lack of strong foundations for the MDA' 4-layers meta-pyramid leads to a common mockery: "So, MDA is just about Egyptology?!". This paper is the pilot of the series called "From Ancient Egypt to Model Driven Engineering". The various episodes of this series show that Egyptology is actually a good model to study MDE.

## 1 Introduction

Model Driven Architecture (MDA) is a recent standard proposed by the OMG [5]. According to this software engineering standard, developing a software is just about developing a series of models expressed in different meta-models. A sketchy summary of the method would be: start from some Platform Independent Models (PIMs), incrementally transform them into Platform Specific Models (PSMs), and end by generating the code. The jargon used here is defined by the MDA standard. Just like other industrial standards, MDA is a complex set of related technologies [6] with a particularly rich set of acronyms.

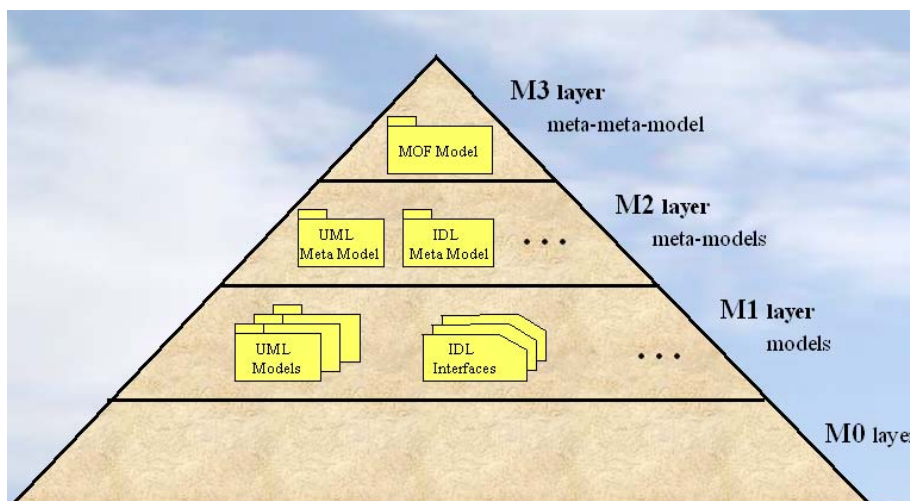


1. Pilot of the series "From Ancient Egypt to Model Driven Engineering" [2]
2. <http://www-adele.imag.fr/~jmfavre>

Acronyms include for instance those for standards (e.g. MDA, CWM, JMI, XMI, UML, OCL, QVT, MOF, SPEM, HUTN) and those for concepts (e.g. PIM, PSM, CIM). At the time of writing this paper, the specifications mentioned above represent more than 2600 pages of documentation. Moreover, each specification is evolving and exists in many different versions. Naturally, only some configurations of these documents are consistent. It is not always easy to determine exactly which configurations are valid, because standards are evolving in parallel and at different rates. Simply put, the MDA set of standards is a quite complex evolving system.

One can wonder how this big pile of documents is organized at the OMG, and what are the exact relationships between these many standards and technologies. One can also wonder what is the strategy behind the production of so many standards. In November 2000, the OMG announced a change from its *Object Management Architecture* (OMA) to its brand-new *Model Driven Architecture* (MDA) [6]. These slogans convey information about of the OMG change of strategy. The former lasted more than one decade. The later is expected to provide a track for the next decade. This change of strategy corresponds to a shift of paradigms. The motto "everything is an *object*" becomes "everything is a *model*" [25].

But what about *architecture*? As pointed out by various authors the last term of the MDA acronym is just a buzzword. In [7], Steve Cook concludes: "*In summary, MDA is a misnamed: It is not an architecture at all*". This is true, at least in the sense that the use of this term has no direct relationships with the notion of software architecture. Though MDA is really a misnamed, the term architecture could however be informally interpreted as a way to organize the elements of the various standards. The following figure shows the famous (yet controversial) 4-layers "architecture" popularized by the UML, MOF and MDA standards. This schema is a coloured version of Figure 2.2 from the MOF specification [8]. It shows that the pile of documents from the MDA standard collections could be arranged in a structure that look like a Egyptian pyramid.



**Figure 1** The four-layers metamodeling pyramid  
A master piece of the Model Driven Architecture (OMG Dynasty, 2000 A.D.)

There is a lot of debate about this pyramid. The foundations of the MDA standards are not clear and this standard received a lot of criticisms. The complexity of this standard, its continuous reference to technical aspects, and its incremental definition, makes MDA particularly difficult to grasp and evaluate. Model Driven Engineering (MDE) seems to be a promising approach to solve industrial issues, but underlying concepts such as models, metamodels or transformations should be studied from a research perspective, and from a broader point of view. In this paper a "mega-model" is introduced as a mean to define the essential concepts and relations of MDE. This mega-model is carefully defined. The mega-model is represented here as a UML model with OCL constraints, though other incarnations of the mega-model in Z, Prolog and Hieroglyphics, are under construction.

The shape of the OMG meta-pyramid recalls the architecture from Ancient Egypt, but this paper shows that the comparison with antic worlds could be pushed much further. This analogy helps in ensuring that the mega-model is technology independent, because the concepts are validated through the use of a wide range of artefacts, from antiquities to modern computer programs. The analogy also helps to get an intuitive idea of what MDE is all about, and this independently from complex technologies such as the MOF for instance. After all, models have been used for ages.

One can learn much by considering an historical perspective. Y.F. Chen, the inventor of the Entity Relationship (ER) data model studied how the concepts he identified in computer science in the 70's, were already present, though in different forms, in Ancient China and Ancient Egypt. For instance, in the paper entitled "From Ancient Egyptian Language to Future Conceptual Modelling" [1], Chen studied the relation between Hieroglyphs and ER modelling. There is indeed a close relationship between the notion of model and the notions of language, syntax, semantics, etc. The language theory, which is undoubtedly a corner stone of computer science, takes its origin in the study of natural languages. Saussure and Pierce were the pioneers in linguistics at the end of the 19th century [3]. Semiotics or semiology, that is the study of signs, is in fact very close to the study of models, because models are ultimately made of signs and symbols. The importance of making the distinction between the objects and the signs that refer to them was already known in Ancient Greece, thanks to the work of Aristotle. As Sowa pointed out, today's technologies and in particular the World Wide Web is based on concepts that come from the Antiquity [4]. Maps are also archetypes of the notion of model. In his work about MDE, Bézivin often refers to maps to illustrate the notions of model and metamodel (e.g. [12]), and interestingly the history of maps is very rich since it runs over millenniums.

Simply put, our goal is to study Model Driven Engineering as practised in Software Engineering, but taking a broader perspective than the one provided by the OMG, whose goal is to promote a particular set of technologies. This paper starts from Ancient Egypt, where pyramids were built successfully more than 4000 years before the MDA pyramid. This paper is indeed the pilot of a series entitled "From Ancient Egypt to Model Driven Engineering". We follow the track of Egyptologists such as Champollion and Joseph Fourier, who were at their time professors at the University of Grenoble. The goal of these papers is to study the foundations of MDE and to a lower extent of Egyptology; with a greater emphasis on the former topic, and a lower accuracy on the latter.

In this pilot episode only one concept is presented, the concept of *model*. It is shown how everything starts from Egypt and from this concept. The remainder of this paper is structured as following. In section 3 explains what is the difference between MDE and MDA, and why it is worth to study MDE. The notion of *mega-model* is presented in section 3 and related work is shortly discussed. The concept of *model* is then introduced in section 4, and section 5 explains what *model engineering* is all about. This first episode ends with a conclusion in section 6, but further episodes continue the series. For instance, *meta-models* are guess stars of episode II [29] and in episode III we will discover what *meta-model engineering* is about [30]. Further episodes describe other essential concepts such as syntax, semantics and transformations. They cover other subjects such as the relationships between metamodels and ontologies, or the relationships between particular technological space and the megamodel presented here. The mega-model is incrementally refined in each episode of the series.

## 2 MDA vs. MDE

Making explicit the foundations of Model Driven Engineering is an ambitious task and it will require a lot of energy. As pointed out by Bézivin in [25], it took a lot of time to reach a common understanding of what are the basic elements of the object oriented approach. Though some points are still sources of debate, at least this field has matured enough to provided "good-enough" industrial-strength solutions. This will certainly be the same for MDE [12] and it will take a lot of time to understand what models are actually and how they can help in building large software systems. In this series of paper the focus is on large scale software evolution [31][45], and toy examples are not considered. But, before to launch the pilot of this series, it is wise to consider why MDE should be studied in the first place. Pros and cons have to be confronted.

### 2.1 Why MDA may fail

Let's start with a short review of the arguments used by MDA detractors and sceptics. Scepticism is a guarantee of a good scientific approach.

The announcement of MDA in 2000 had in fact very little impact on the research community. In the last years however, the situation changed and today MDA has some supporters but also many detractors. MDA is indeed subject to a lot of criticism in the research community. Though the MDA pyramid makes sense from an industrial point of view, its exact structure has been a source of debate. For instance, the M0 layer is subject of much controversy (note that in [8], the M0 layer was left blank in Figure 2.2, but filled with an example in Figure 2.1 of the same document). The lack of strong foundations for the MDA architecture leads to a common mockery:

(I) "*So, MDA is just about Egyptology ?!?"*"

Though the word "pyramid" is not used at all in the OMG specifications, as said before references to Egypt makes sense, at least at the symbolic level. The figure below shows the Saqqara pyramid, which is known as the first stone building on Earth. The layered architecture of this pyramid is clearly apparent.



**Figure 2** Saqqara "step" pyramid (2nd Dynasty, 2630 B.C.)  
First stone building on Earth, a master piece of Ancient Egypt architecture

As this photography suggests, the numbering of levels depends on the relative position from which the pyramid is observed. It will be shown in [29] that this property applies to the MDA architecture as well, and that pyramids provide indeed a good models to understand the MDA approach.

MDA is sometimes announced as the next paradigm in Software Engineering. Some marketing presentations even suggest that it could be the next silver bullet. Some even claim that this approach will revolutionize Software Engineering. These kinds of statements often leads to the following comment:

(2) *"There is nothing new in the MDA! Similar approaches had existed for long."*

The MDA approach is based on the use of abstract representations of software that are called "models", and from these models the idea is to produce the code. In other word, raising the level of abstraction is one of the core idea of MDA. But, that's also the aims of almost all existing techniques in Software Engineering. Using the term "model", instead of "abstraction" or "specification" will obviously not to be enough per se for a revolution in Software Engineering.

The notion of "platform" also plays an important role in MDA. The idea is to build models that are independent from the platform. But after all, what was the idea behind programming languages and compilers? It was precisely to produce computing models independently from hardware architectures and micro-processors. Obviously, there are necessarily connections between MDA and compilation techniques. Introducing the notion of platform is not enough to radically change Software Engineering.

Another important characteristic of MDA is the use of meta-models and metamodelling techniques as a mean both to describe languages and to structure software artefacts. Again, there are plenty of techniques in computer science to describe languages, starting from the huge work on syntax and syntax-directed tools in the late 70's and 80's. One can argue that MDA is about managing graphs, not only trees. But there has been also a significant amount of research on graphs and graph manipulation.

The idea of making explicit the relationships and structures of software artefacts is far from new. Just remember large software engineering projects such as AD-lifecycle from IBM, and the PCTE standard in Europe. Building a library of meta-models or schemes to promote reuse and integration is not new either.

The novelty does not come either from the MDA pyramid. Similar structures were present in standards such as IRDS, PCTE, CDIF, ODMG'93, etc. (e.g. [13]). What is more, the Saqqara pyramid was built more than four thousands years before the MDA pyramid. In that condition it would be difficult to argue that this architecture is new...

Another concern about MDA is about its complexity and quality of its foundations.

*(3) "Concepts in the MDA standards are unclear, ill-defined and too complex"*

The MDA set of standards comes from industry. It does not attempt to reach the perfection. The challenge faced by industrial standards is instead to get to be adopted, used and supported by a large community of software developers worldwide. These standards are defined incrementally and their history often reflects the series of confrontations and consensus that animate contributors. It should not be surprising in such conditions that the quality of the standards does not compare favourably with well-thought software engineering techniques such as formal specifications.

## 2.2 Why MDE may succeed

Formal specifications languages such as Z, VDM or B had received however little attention from industry. Despite the advantages brought by these techniques, software development community does not seem to be mature enough. Moreover, it is not clear that the formalization required by these techniques is worthy in common software engineering projects; at least when considered with respect to the additional cost implied.

This contrasts with the UML industrial standard which is becoming more and more popular, although it has received a lot of criticisms. Despite UML lack of rigor, in many situations some its subsets are considered as "good enough" to fulfil typical needs. More importantly, UML has drawn the attention of developers to the fact the source code is not the only way to think about software. For instance class diagrams are now considered as practical means for developing and understanding software. Simply put, UML has popularized the notion of model in industry; or at least it has provided programmers a first understanding of what a model could be.

In fact, the XML standard has played a similar role. It becomes each day more obvious to programmers that software is not only made of code. Software also includes XML files, databases, UML models, and so on. Improving a database schema or defining some XML DTDs are now considered as natural activities when programming. The increasing importance of techniques such as introspection, when dealing with component programming or distributed computing, also leads to the popularization of meta information and meta-levels. Meta-models are slowly finding their path to industry, while this concept was considered as quite obscure a few years ago. Code generation also becomes a common practice with IDEs. Today no professional programmer is surprised to see an XML file transformed into code. The availability of commercial code generators and associated tools also put emphasis on the need of customizing such transformations. That's exactly, what the MDA is intended for.

As suggested by Uhl, it seems that "Model Driven Architecture is Ready for Prime Time" [14]; at least on software industry channels where MDA arouse a growing interest. The complexity of technological platforms such as J2EE or DotNet is ever increasing. This gives rise to new challenges in software engineering. Though concepts such as those provided by PCTE where certainly good, they certainly came to early, in a market which was neither mature enough to received them, nor demanding enough to study and adopt them.

This is the same for syntax-directed technology which didn't found the place it deserved in software industry. By contrast, XML has become extremely popular thanks to its simplicity. As said before defining new syntaxes, that is new DTDs and XML schemes, is now commonplace.

The same phenomenon can also be observed for transformation systems. Those provided by research projects had a limited impact in the past. Today transformation languages such as XSLT are used daily in web development. All these concepts and techniques are now an integral part of the software development landscape. And this will not change in a near future.

It makes no doubt that the MDA set of standards suffers from many deficiencies. Though MDA, as defined by the OMG, is a set of a specific industrial technologies with their own issues, model driven approaches seem however to be very promising. And software industry is demanding for solutions.

After a period of relative silence from 2000 and 2002, the interest starts to grow in the research community. For instance, since 2003 many conferences and workshop started to included in their calls for papers references to model-driven approaches. Though the MDA standard has been seminal, the current trend is to consider it just as an step towards of a more general approach called *Model Driven Engineering* (MDE). In particular *Model Driven Software Engineering* (MDSE) is the intersection between MDE and software engineering, that is it is the subset of MDE which is concerned with software production. Various research projects have been launched over the world to study the potential of MDE and to define the underlying concepts.

### **2.3 Modelling and Egyptology, two French Traditions**

There has been a long tradition in modelling in France. For instance, the model-based specification languages such as Z took one of their roots in the early 70's in Grenoble, when Abrial defined the first version of Z. B, its successor, is also well supported in France. Now, various french companies are also quite active in the domain of model driven engineering [15]. Similarly various research projects and initiatives has been launched by national research institutions. For instance, the OFTA group recently published a survey on MDE [16]. This survey covers both industrial issues and research topics [15]. S. Gérard and P.A. Muller has launched an initiative called TopModL [19]. TopModL is an open consortium for the development of an open source MDE environment. National organisations for research such as INRIA also fund research projects to study MDE. In particular research on model transformation is quite active in France [17].

In July 2003, the CNRS national institute for research also initiated a research project which aims at establishing a map of MDE research issues and opportunities. An explicit goal of this consortium is to establish which are the foundations of MDE, irrespective of a particular technology or standard. This should bring an answer to criticism #3 mentioned above, that is the lack of foundations for MDA. Another core activity of the group is to study how this field is connected to other areas in computer science. MDE is best seen as a synergy between existing work. This is a direct answer to criticism #2, that is the lack of references to previous work.

But what about #1? MDA is about Egyptology, and Egyptology should not be turned into derision. The French were the first to consider Egyptology as a very serious matter. Egyptology was studied for the first time from a scientific point of view, with a systematic approach during the campaign of Egypt which started in 1798. Napoleon had brought with him between nearly 1000 civilians including 167 of whom were scientists, technicians, mathematicians and artists who studied the art, architecture, and culture of Egypt. From 1809-1828, they published a 19-volume work called *Description of Egypt* [21]. Their observations, drawings and illustrations were circulated throughout Europe and created a tremendous interest in antiquities of Egypt. This is in this context that the French scientist Joseph Fourier (1768-1830) went to Egypt. Nowadays, his numerous contributions in mathematics are well known, but Fourier also wrote various reports about Egyptology [20]. Joseph Fourier also gave later his name to the University of Sciences in Grenoble. At the same period Jean-François Champollion (1790-1832) and Jean-Jacques Champollion (1778-1867) had also been professors at Grenoble. There were both recognized scientists and Egyptologists. In particular, Jean-François Champollion is acknowledged as the father of modern Egyptology thanks to his work on deciphering the Hieroglyphics. The story of the Rosetta Stone is related in [30] and it is shown how Champollion laid the foundations for both of Egyptian archaeology and of metamodel reverse engineering.

## 2.4 Summary

This section can be summarized by the following statements.

- Model Driven Engineering (MDE) is a global software engineering approach.
- MDE aims at integrating existing results and bodies of knowledge.
- There is nothing radically new in MDE, except may be this integrative approach.
- MDE must not be confused with MDA.
- Model Driven Architecture (MDA) is a specific standard from the OMG.
- MDA specification has deficiencies and that's why research is required.
- MDE comes after the failure of various projects such as AD-cycle, PCTE, etc.
- MDE should learn from past experiences.
- Software industry seems to be mature enough to receive MDE solutions.
- Software industry is demanding for MDE solutions.
- It will take time before reaching a common agreement on what is MDE exactly.
- The concept of model had been used for ages.
- Egyptology is a French tradition.

The series "From Ancient Egypt to Model Driven Engineering" could indeed be seen as an answer to MDA detractors. Firstly, it is true that MDE is about Egyptology. Secondly, it is true that there is nothing really new in MDE. That's precisely what makes the power of MDE. Finally, it is also true that the foundations of MDA are not clear. That's precisely why more research is required. Model Driven Engineering is an approach that seeks to define a strong method for software development by integrating into a common framework well-established techniques.

### 3 Towards a Mega-model for Model Driven Engineering

One of the issues with the MDA standard is that there is no clear separation between essential concepts and the technologies that implement these concepts. Remember that the MDA is not a research proposal, but an evolving industrial standard. A lack of rigor characterizes the specifications, which are not anyway assumed to be perfect.

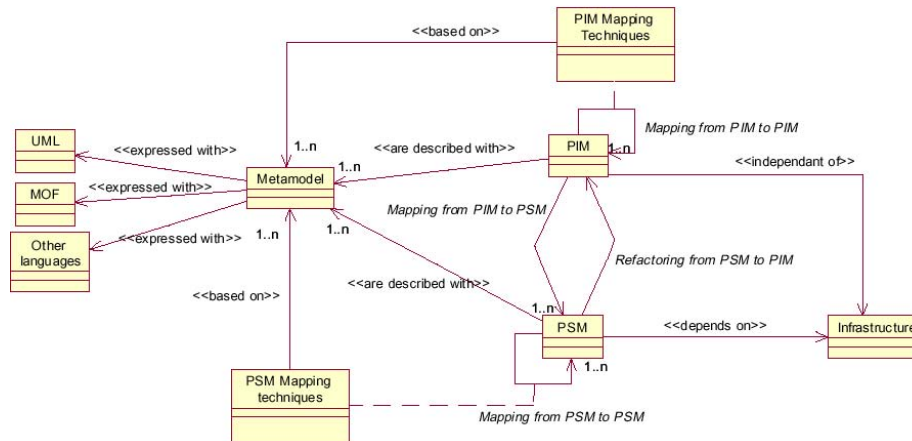
For instance, the elements of the various pyramid layers are usually assumed to be linked by so-called "instanceOf" relationships. While this could be true in the object-oriented context, this relation is not recognized as such in other contexts such as grammars and languages, databases, XML technologies, etc. This is a problem in the context of MDE, since this approach aims at integrating various technologies, not only one. In fact, due to its root in OO technology, everything tends to be called Object, Class, Inheritance or InstanceOf in the context of the OMG specifications. These terms are often used quite informally. Sometimes they describe situations that actually correspond to different concepts [24][25][26]. Similarly, a common mistake is to define a meta-model as being "a model of a model". Though the notions of model, metamodel, platform, PIM and PSM are central to the MDA standard [53], these concepts are poorly defined.

This series of papers relies on the work of Seidewitz [23] and Bézivin (e.g. [25]) as references frameworks for understanding the foundations of MDE. While in [23], Seidewitz describes informally, yet thoughtfully, concepts such as *models* and *meta-models*, in [25] Bézivin identifies two fundamental relations coined *RepresentationOf* and *ConformantTo*. There are also other frameworks around bringing other perspectives on MDE. This is the case for instance for the framework provided by Atkinson and Kühne [24] which clarifies the relationship between meta-model and ontologies.

In [31] a framework modelling the software space as a 3D space is introduced to clarify the notion of meta-model and model co-evolution. In [35], a broader vision is presented: the framework not only describes the concepts of model and meta-models, but also the concepts related with (meta) modelling methods and procedures.

In the context of this paper these frameworks are called *mega-models*. Though technically speaking a mega-model is a meta-model, this term is not used to avoid confusion; in particular because metamodels, which are elements of the MDE domain, will naturally fit as an element of the mega-model. Simply put, the idea behind a mega-model is to define the set of entities and relations that are necessary to model some aspect about MDE. In the context of this paper, we call "mega-model" a model of MDE. The problem then is how to define mega-models. While this can be done in plain English, this would lead to many ambiguities and this is what we want to avoid. The mega-model presented in this series will be expressed in UML with OCL constraints.

Note that the usage of UML for such a purpose is not new. For instance the following figure comes from the MDA specification [5].



**Figure 3** Example of a very informal mega-model ([5], page 12)

As it can be seen, this diagram introduces concepts such as PIM, PSM, Metamodel, Infrastructure, Mapping, etc. This sketchy model is not different from the text it goes with. A closer look at the diagram shows that there are many different associations and stereotypes; that are used in a very informal way; that cardinalities and navigation directions are somewhat strange. In fact, this kind of mega-model is too informal to be really useful. This mega-model is not even used in the rest of the document [5], and the few examples that follow this diagram do not attempt to be conformant to the mega-model.

In this series, the goal is much more ambitious. What is needed is a mega-model that is good enough to reason about MDE with confidence. Each element of the mega-model will be carefully introduced and many examples will help to test the mega-model. We strongly believe that a common agreement could not be reached on the meaning of a concept, if not enough examples make it possible to validate or invalidate a given statement. The goal of each example in these series is to enable the reader to check whether he or she agrees with the definitions provided.

According to Bézivin, the object-oriented approach is organized around two elementary associations, namely `InstanceOf` and `Inherits` [25]. In other words, the quality of a mega-model should not be measured by the complexity it introduces, but on the contrary by its conciseness and power. The lesser the better.

Based on existing frameworks, we are in the process of defining a mega-model in the context of the French national project AS-MDE [18]. In this paper only one association is introduced, because this first episode of this series is entirely devoted to a single concept: models. Other episodes gradually introduce other concepts. For instance the notion of meta-model is introduced in [29]. Finally, note that, as any other models, mega-models are not expected to be perfect, but just "good-enough" for a given purpose. Like any other model, a mega-model is the result of an iterative process, and as such it is subject to continuous improvement. The reader is invited to consult the website of this series [2] for up-to-date information on this subject.

## 4 Models

Though the so-called model-driven engineering is considered as a modern software engineering technique, the notion of model is indeed very old. Museums are full of ancient models. Museums actually play the role of *models repositories* [28].

For instance Figure 4.a shows a model that represents the pharaoh Tutankamon. The photography is itself a "model" of the physical model kept in the Museum of Cairo. The relationships between models will be studied in section 4.2.

Similarly it is clear that Figure 4.b is also a model. It is believed that it could either represent Nivizeb, the father of Ancient Model Engineering, or another priest called Wendjebauendjed. There is still some controversy about this subject, but it is established that Figure 4.b represents a great priest.

Note that (a) and (b) are both models of humans for the usage of gods. On the contrary, Figure 4.c is a model of a god for the usage of humans. This is a statue of Thot, the god of Moon represented either as a Baboon or an Ibis. Thot is assumed to be the inventor of speech and script. As it will be related in episode III [30], Thot is who has given meta-modelling to mankind.

Figure 4.d is a photography of a small fragment of the Fidus Papyrus as it was before its restoration at the Museum of Grenoble. As it will be related in section 4.2, the Fidus Papyrus is a model of the Solarus (Figure 4.e), which is itself a very ancient model of the solar system (section 5.7). In fact as it will be explained later, Figure 4.e is only a small piece of a model, and it is still not clear if it is itself a model or not. Similarly, Figure 4.f is not a model, but a model repository. This small chest is made of ivory and gold, but more importantly it contains a set of very ancient meta-pyramid drawings by Nivizeb the priest as related in the Story of Thotus the Baboon [29].

The Dagktis Stone is represented in Figure 4.h. This model is kept in the basement of the Museum of Grenoble. It models what happened in the so-called "Dagktis seminar" [28]. Finally, Figure 4.h is the famous Rosetta Stone kept in the British Museum. The story of the Rosetta Stone will be related in [30]. Champollion received a model of the Rosetta Stone and deciphered this model in 1822.

Though the notion of model is very old, a rigorous definition is required. The next section introduces a very simple mega-model referred as the  $\mu$ -MegaModel because it is made of only one association called RepresentationOf,  $\mu$  for short. Note that though models were used for long in ancient times, the notion of meta-models was usually not present, at least explicitly. This paper put the emphasis on the fact that the notion of model could be totally separated from the concept of meta-models. The notion of system is in fact all is needed.



(a) Model of Tutankamon  
(Museum of Cairo) [54]



(b) Model of Nivizeb [54]  
(Museum of Cairo)



(c) Model of Thot [54]  
(Musée du Louvre)



(d) Fragment of the  
Fidus Papyrus  
(Museum of Grenoble)



(e) Fidus Bone  
(M. of Grenoble)



(f) Nivizeb's pyramid models  
(Museum of Grenoble) [54]



(g) Dagktis Stone  
(Museum of Grenoble)



(h) Rosetta Stone  
(British Museum)

**Figure 4** Examples of ancient egyptian models from various model repositories

## 4.1 Systems

Let's start with a very abstract definition:

*A system is the primary element of discourse when talking about MDE.*

This definition which seems too abstract to be useful, will make more sense when combined with other concepts described in the remainder of this paper. Before introducing these concepts, let us however introduce a rough classification of systems. Three categories are of interest here: physical systems, digital systems and abstract systems.

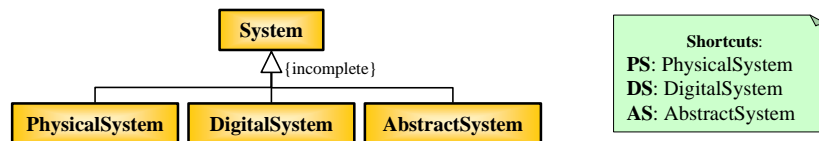


Figure 5 MegaModel: System

*Physical systems (PS) are observable elements or phenomena pertaining to the physical world.*

*Digital systems (DS) are those systems that reside in computer memories and are processed by computers.*

*Abstract systems (AS) are ideas and concepts that eventually reside in human mind to be processed by human brains.*

This separation in three categories is somewhat arbitrary, but it corresponds to three different "technological spaces" that will be used later in this paper (section 5.2). Anyway, what is important right now is that this classification is good enough to represent typical situations. For instance let's consider Fido and Lassie, the dogs traditionally used as guinea pigs in various domains (e.g. [34]) and in particular in the meta-modelling literature (e.g. [24][27]). Fido and Lassie are physical systems. The photos on the previous page depict physical systems. The solar system is another example of physical system. A set of spheres is an abstract (mathematical) system. Databases and programs are digital systems. The following figure depicts a UML object diagram that model these particular examples. This diagram is conformant to the mega-model and to the UML notation. This approach is used throughout of this series to provide examples.

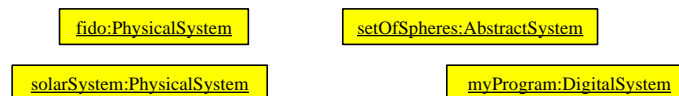


Figure 6 Examples of systems

One important feature of systems is that they are enough complex to be studied and that they can usually be decomposed in sub-systems. For instance the solar system is made of planets; a sphere can be decomposed in its surface and its volume, a program can be decomposed in procedures, types and so on. Physical systems have the property to be virtually infinite, while digital systems are definitively finite systems. In fact, the decomposition of systems into sub-systems is a fundamental relation that will be discussed in another episode. This paper concentrates deliberately on a very simple mega-model. The following section describes the concept of model.

## 4.2 Models, Systems Under Study and RepresentationOf ( $\mu$ , )

Instead of providing our own definition of what a model is, let's cite some existing definitions. In the context of the UML standard, the term model is defined as following:


"A **model** is an abstraction of a physical system, with a certain purpose.

As discussed after, we don't agree with the restriction to *physical* systems, but anyway let's consider the definition provided by Bézivin and Gerbé.

"A **model** is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the **actual system**." [26]

Seidewitz provides another definition:

"A **model** is a set of statements about some **system under study (SUS)**." [5]

From these definitions we can at least identify three notions: the notion of model, the notion of system under study (SUS) and a relationship between these notions. In [25] this relation is called RepresentationOf. It is called Represents in [24], and the inverse relation is called Describes in [53]. In Ancient Egypt, this relation was symbolized by a crocodile  [30]. In this paper it will be noted for  $\mu$  short. Whatever its name or the corresponding symbol, it is very important to understand that this relation is actually defined between systems: being a model or SUS is a relative notion, not an intrinsic property of an artefact. In fact these notions are roles that a system can play with respect to another system as depicted in the following mega-model class diagram. The role played by a system actually depends on the use of this system as illustrated by the following story.

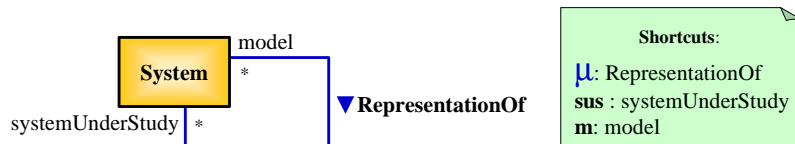


Figure 7 MegaModel: RepresentationOf ( $\mu$ , )

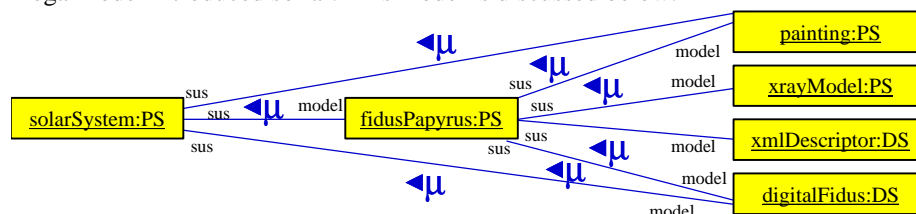
Antonio was an egyptology enthusiast. He went to Egypt to visit the pyramids with his faithful dog, Fido. During a walk with Antonio, Fido found an old papyrus near to the ruins of a very ancient pyramid. From his point of view the papyrus was just a physical system. When Antonio saw Fido playing with it, he immediately removed it from his mouth. With a closer look he saw what seemed to be an ancient map, or at least something similar. In the same day Fido found a really huge Bone. Antonio didn't want either Fido to play with it because the bone wore at one end inscriptions in gold lettering. Back to home, Antonio brought the papyrus and the bone to the Museum of Grenoble (Figure 4.d. and Figure 4.e) Nobody understood the purpose of the bone, but some after a while an historian from the University of Grenoble discovered that the papyrus was actually a ancient model of the solar system<sup>1</sup>. The papyrus and the bone were donated to the museum and named "Fidus Papyrus" and "Fidus Bone" after the dog. The Fidus

1. This was just an hypothesis, because nothing linked the model and the SUS.

*Bone is now displayed in the Museum of Grenoble (Figure 4.e). By contrast, the Fidus Papyrus was so deteriorated (because of the effects of the ages and of the jaws of Fido), that it was impossible to display it to the public. The museum decided that the papyrus deserved a restoration and required further studies.*

*First an artist considered the Fidus Papyrus as a system and built a model of it, in the form of a painting on a new papyrus. This copy was later exposed in the museum. Though imperfect, this model was considered by the Guide of the museum as "good enough" to explain to the visitors how the Fidus Papyrus looked like. After being received by the museum, an XML file modelling the papyrus was produced for the usage of the archive system. Though the XML descriptor only included information about the origin of the model, its dimensions, and a short description, this file was considered as good enough for archive purposes. It was enough to answer every question the archivist could have. The Fidus papyrus was later studied in a laboratory where a technician decided to make an X-rays photography. Using the X-rays technology was indeed a good idea, because the X-rays model revealed some traces of ink that were no longer visible on the Fidus Papyrus (though there were obviously there). The Fidus papyrus became so popular that a digital model was produced and sold in Museum store under the reference Digital Fidus. Using software technology brought many benefits since it enabled visitors and clients to manipulate the model very easily. It made it possible to see the Fidus Papyrus both as it was found and as it was supposed to be before the ink disappearance and Fido's game. At this point of this story, nobody knew that the mysterious author of the Fidus Papyrus, was Nivizeb the priest. And nobody knew either the secret of the Fidus Bone.*

Let's summarize the beginning of the story by means of a model expressed with the mega-model introduced so far. This model is discussed below.



**Figure 8** Model of the Fidus Papyrus story ( $\mu$ -graph)

The story shows that the role played by a given system depends on the context in which it is considered. The papyrus was just a (physical) system in the jaws of Fido. It became a model in the hands of the historian. Finally it was considered as a system under study (SUS) in the hands of the artist, the archivist and the technician. This difference of status depends on the usage of the system. Fido just wanted to play with the papyrus. The historian wanted to interpret it<sup>1</sup>. The artist, the archivist and the technician

1. The relationship between the interpretation relationship and models is further described in [23]. It will be described later in the series because it implies adding the decomposition relation ( $\delta$ ) to the megamodel. Similarly, the actor who interprets the model is also an important aspect of a model. The study of this concept, on which semiotics provide some insights [3], is also postponed to a later episode.

wanted to studied it or describe it. Since they had various concerns and considered a different set of questions, they produced various models. Each model was considered as "good enough" for a given purpose. The role played by of the Fidus Bone could not be determined.


As the story also suggests, it is quite common to compose  $\mu$  links, in other words to form models of models, models of models of models, etc. For instance, the painting is a model of the Fidus Papyrus which is itself a model of the solar system. A visitor taking a photography of the painting will naturally produce a model of model of model, and so on. At this point emphasis should be put on the fact that a model of a model is *not* a metamodel (this notion is defined in Episode II [29]).

A model of a model of a SUS is in some cases a model of this SUS, but in other cases it is *not*. In other words the RepresentationOf relation is not transitive, although this transitive behaviour often applies between particular models. As an illustration, the painting can be considered as a "transitive" model of the system solar. By contrast, this is not the case for the X-ray model or the XML descriptor. For instance, while the dimensions of the papyrus stored in the XML descriptor is of interest for archiving purposes, this information is by no means related to the solar system.

### 4.3 Summary

Summing up, this section has shown the following properties.

- A system can play the role of model with respect to another system referred as the system under study (SUS).
- A model is an abstraction of a system. It is good enough for a given purpose.
- The relation between a model and a SUS is called RepresentationOf,  $\mu$  for short.
- The notion of model is relative. This is *not* an intrinsic property of a system.
- Models can be systems of arbitrary kind.
- Systems of arbitrary kind can be modelled.
- A model of a model is *not* a metamodel.
- $\mu$  is not transitive, but often a model of model of a SUS is a model of that SUS.
- To define what is a model, it is not necessary to define the notion of metamodel.

The mega-model introduced so far is very simple. It is only made of 4 classes: System, PhysicalSystem, AbstractSystem and DigitalSystem, and one association RepresentationOf ( $\mu$  or ).

## 5 Model (Driven) (Reverse) Engineering

While the mega-model introduced so far is good enough to describe fundamental links between systems, it does not provide any hint about the actual usage of the models. In particular the intent behind the  $\mu$  links is not captured by the RepresentationOf association and no particular method has been assumed for the production of models. Understanding these aspects is necessary to understand *model engineering*.

### 5.1 Models and Separation of concerns

The first question to be answered is who use models and for which purposes. Obviously, the answer depends on the engineering discipline. However in each case *separation of concerns* is an underlying principle. When considering a single system, different people want a different set of questions to be answered. Separation of concerns is an intrinsic property of model engineering leading naturally to aspect-oriented modelling. From an organizational point of view, separation of concerns can also lead to the separation of jobs, that is to the definition of a set of actors with well identified roles and skills. For instance, the painting is used by the Guide of the Museum to relate the story of the Fidus Papyrus. The XML descriptor is used by the archivist, the X-rays model by the technician, etc. All of these actors have different skills and concerns.

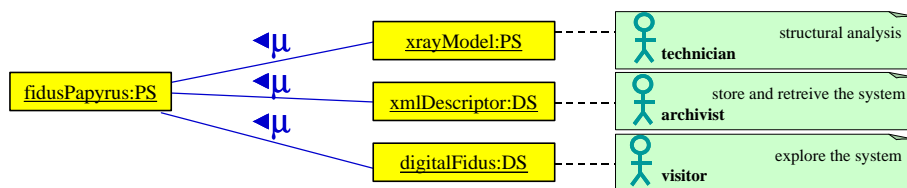


Figure 9 Separation of concerns

### 5.2 Models and Technological spaces

While the separation of concerns implies considering different sets of questions, *Technological Spaces* (TS) enable to answer the same question but using different technologies. Not all technological spaces are equivalent. Some questions can be easy to answer in some TS, but difficult in others. The notion of technological space has been introduced in [33]:

"A **technological space** (TS) is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities."

In [33], the authors divided the digital space in various technological spaces. This includes for instance Documentware, that is technology for structured document (e.g. XML), Grammarware, that is technology based on grammars [40], Dataware and database management systems, Ontologyware and ontology engineering, Modelware and model-based technology (e.g. UML), etc. Whatever, their boundary and their structures, the existence of various technological spaces means that given a system, one has to choose the TS that will be most appropriate for the expression of a model or a given usage. Abstraction should not remove the fact that reality is driven by many competing technologies. Another description of the software space which link abstract and concrete views is given in [31]. Here, a broader perspective is taken. The discussion is not restricted to software models. For the sake of simplicity, the distinction has been made between physical models, abstract models and digital models (see section 4.1).

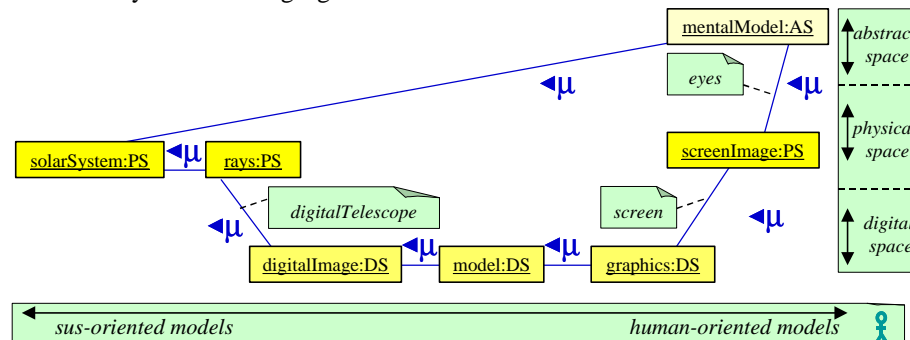
*When Nivizeb wanted to make some computations, sometimes he just applied mental arithmetic, but if the computation was more complicated he used a kind of abacus of his own invention. By physically moving stones and bones in the sand, he reproduced arithmetic operations. Obviously, if he had a computer, he would transport his problem into the digital space and use software models.*

Moving from a technological space to another is a common operation [33]. The following figure shows the chain of steps that leads from the solar system to the mental model of the solar system in the brain of a visitor using the Digital Fidus.



**Figure 10** Example of chain of models across technological spaces ( $\mu$ -graph)

As illustrated by the figure, the  $\mu$  links often cross technological spaces. Another important property of technological spaces is that they are no such thing such as the "best technological space". The choice depends on the problem at hand. Fortunately "no TS is an island and they are bridges between spaces" [33]. Changing from one space to another is therefore important. That means that the *import* and *export* operations to enter and leave the space should be facilitated. Note, that the interface between the physical space and other spaces is typically achieved by means of input and output devices as illustrated by the following figure.



**Figure 11** Chain of SUS-oriented models to human-oriented models ( $\mu$ -graph)

This figure shows a typical pattern in which a problem that can be solved in one space but projected into another space to improve some feature such as precision, efficiency, etc. For instance, let's assume that Antonio want to observe the solar system. Antonio, who would be on the right of the figure, could directly look at the solar system on the left. Instead of that Antonio could use a computer connected to a digital telescope. This complex system will provide Antonio with an improved model. A digital camera is used to import some physical phenomenon into the digital space, then a digital chain of models are produced and eventually exported back to the physical (and observable) space through the screen. Eventually Antonio eyes and brain process the information.

As the figure suggests, models can be roughly organized along a dimension that go from *SUS-oriented models* on the left to *human-oriented models* on the right. For instance, in the digital space XML models are used for computer manipulation while HTML models are used for presentation purposes. Though the chains of models could be rather complex, the human at the end of the chain often constitutes a bottleneck. The relationships between the SUS, a model of this SUS and the mental models produced these systems when observed by a human have been studied in the field of semiotics [3].

In the context of this first episode, it is enough to consider how amazing is the ability of the brain to perform short-cuts when it encode/decode information. For instance, when the visitor look at an image produced by the Digital Fidus he can think directly in terms of the solar system. That means that most of the time, some intermediate levels of models are often omitted in chains of models. Models are often mixed up if they are close. For instance, in the figure above screenImage could have been omitted. Various models on the left have been indeed omitted. The digital telescope is certainly a quite sophisticated systems that produces many intermediate models. It gathers information through a set of lens; other mechanisms process the resulting image, and so on. The natural omission of some "intermediate" models explains in part why reasoning about models is often difficult: very often in discussion about models one argue that some model is missing or that another one is not needed, etc. For instance when speaking in terms of a map of the solar system, one can think in terms of a printed map, or a digital map in a computer or on the screen, or a mental image. When the  $\mu$  links have a transitive nature, confusing these systems is usually safe. Semiotics on the contrary to distinguish each levels, including mental model produced by symbols [3][4]. But this is another story.

Figure 11 also suggests that modern system engineering implies actors from various engineering discipline. For instance building a telescope system imply many different skills ranging from physics, mechanics, mathematics for the telescope part, computer engineering for the digital part, cognitive science and ergonomics for the interface with the user, etc. Technological spaces often have an influence on organizations, because each TS required special skills. Software engineering focuses on the digital space. This space is further refined into sub-spaces in [33]. As pointed out by the authors an important research issue is to draw a map modelling existing technological spaces and sub-spaces, as well as bridges between these spaces.

### 5.3 Specification Models vs. Descriptive Models

Model engineering means producing model, but what for? Models can be used either to *specify* a system to be built, or to *describe* an existing system. This leads Seidewitz to introduce the distinction between *specification models* and *descriptive models* [23]. This distinction has also been introduced in the revised version of OMG' MDA Guide, and this in very definition of the notion of model:

*"A model of a system is a description or specification of that system and its environment for some certain purpose" [22].*

Simply put new systems are produced from specification models, while descriptive models are produced from existing systems. Making the distinction between specification models and descriptive models is useful to express "who, of the model or the system, have the truth"<sup>1</sup>. In the case of specification models, the model holds the truth: a

---

1. In fact the separation between specification models and descriptive models is not always clear cut. A more complete treatment of this topic would required to introduce the notion of model/system co-evolution. The simplification provides here is "good enough" for the purpose of this paper.

SUS is said to be *correct* according to a specification model, if every properties defined by the model is satisfied by the system produced [23]. In the second case, the system represents the truth. A descriptive model is said to be *valid* if everything said by the model about the system is actually true.

*In fact, Nivizeb knew very much about the usage of models. During his life he had produced a huge amount of specification and descriptive models. For instance, when he designed some pyramids (see the story of Thotus the baboon [29]), he produced specification models. Nivizeb was very demanding. He warned everybody: the pyramid had to be conformant to what the models said, else Nivizeb would considered the pyramid "invalid". And he would feed his crocodiles with the slaves. On the other way around, Nivizeb also produced a lot of descriptive models, such as the Fidus Papyrus. This model revealed later to be "incorrect". Nivizeb had however a peaceful life.*

Note that the separation between specification model and descriptive model could be added in the mega-model by introducing two associations named `PrescriptiveRepresentationOf` and `DescriptiveRepresentationOf`. It should be clear however, they would not be totally new associations but just specialization of `RepresentationOf`.

#### 5.4 Platform Independent Models vs. Platform Specific Models

One could think that model engineering is just about (1) producing a specification model, and (2) generating a valid system from that model. This is more complex. Various steps are required, with one or more models being produced at each step. The MDA Standard put emphasis on the distinction of *Platform Independent Models* (PIM) and *Platform Specific Models* (PSM) [5][6]. Sometimes the concept of *Computational Independent Model* (CIM) is also introduced. Roughly speaking a CIM just describes concepts related to a particular domain, but with no reference to the particular problem to be solved in that domain. PIM describes a particular system that solve a particular problem but in a technology independent manner, while a PSM describes how this system can be implemented using a given technology. Though quite intuitive at the first sight, the notion of platform, PIM and PSM are however poorly defined in the MDA standard. They are indeed subjects to controversy (e.g. [7]). There is clearly a continuum between PIMs and PSMs and the distinction between these models is not clear-cut. These notions are relative. At the time of writing this paper we do not believe that these concepts should be included in the Megamodel.

#### 5.5 Conceptual Models, Specification Models, Implementation Models

While the distinction PIM/PSM is not fundamental, it is clear however that any development process will define different levels of models and end with the production of a system. Recognizing the existence of such levels is important. To this end, Fowler suggests in [9] another distinction based on 3 levels of models, namely *Conceptual Models*, *Specification Models* and *Implementation Models*. Simply put *Conceptual Models*, the more abstract ones, describes concepts rather than solutions. They are closed to CIM. *Specification Models* are used to specify the system to be built but without giving details about its actual implementation. Finally *Implementation Models* describe how systems have to be implemented. Though specification models are close to MDA PIMs and implementation models are close to implementation models, these equivalencies do not

hold because the two classifications are based on a slightly different perspective. Fowler put the emphasis on the fact, that each kind of models can be described using the same modelling language [9].

### 5.6 Sketchy Models, Blueprint Models, Executable Models

Another classification of models is provided by Mellor and his colleague in [10] taking yet another perspective on models. The distinction is made between three kinds of models, depending on their usage. A model can be considered as a Sketch, as a Blueprint, or as an Executable. A *Sketchy Model* is not precise or complete, nor is it intended to be. The purpose of such models is typically to try out an idea when the model is a specification or to simplify communication and understanding when the model is descriptive. Usually sketchy models are volatile, they are neither maintained nor delivered. By contrast *Blueprint Models* are more precise and can be used as specification to build a system. What they say about the system is expected to be true, though they describe only some aspect of this system. *Executable Models* are the last kind of models. By contrast to blueprints, they contain enough information to be directly interpreted by a processor or to derive an executable system. Currently this last kind of model is not widely spread. It corresponds to ongoing work such as Executable UML [11] and other MDA approaches. Though not fundamental, this classification recognize the fact, that there is a great variety of models and that a model should be considered as "good enough" for a given purposes. For instance, a sketchy model may sometimes lies about a system, but this is not necessarily considered as a problem.

### 5.7 Illustration: Story of the Solarus

Let's come back to the story of Fido to illustrate some of the concepts described above, the emphasis being put this time on the engineering process of models as practised by Nivizeb the Priest.

*The day Fido found the papyrus, he disappeared during about one hour. Antonio was looking for him. In fact Fido managed to entered in the very insides of the pyramid through a small hole. When he arrived in a small room, fido found a bone [35]<sup>1</sup>. Fido didn't notice that this big bone (Figure 4.e) was a just a sub-system of a strange arrangement made of pieces of wood, carved bones, crocodile teeth and coloured stones. Fidus went out with the big bone. After a while he found Antonio who look at him furiously. Fido gave him the bone. Nobody discovered the secret of the Solarus.*

*By regularly observing the sky Nivizeb had build a classification of planets [30]. He wrote some papyrus explaining what where the core entities along with their respective properties. These papyrus were mostly conceptual models and computation independent model.*

*Nivizeb was puzzled by the movements of the planets and concentrated on predicting their relative positions. He was in particular interested in computing the date of eclipse and other alignment of planets, because he thought that discovering this mystery would gave him a lot of power. After some years Nivizeb had built a simple yet "good*

---

1. The sentence "Fido found a bone" is studied in great details in [35].

enough" mental model of the solar system. He thought about the planets as being just like a set of discs that move around, some constraints governing their relative positions. Since dealing with this mental model was too complicated, Nivizeb decided to built an instrument called the Solarus. The aims of the Solarus was to reproduce the constraints between the position of the planets in the sky.

Building the Solarus was a challenge for Nivizeb. At that time he never had to build a so sophisticated model. He decided therefore to first produce a series of papyrus to specify the Solarus. Before thinking in terms of the Solarus as physical system, he first produced the papyrus which is now known as the Fidus Papyrus (see the top of Figure 12.a). This model just described what Nivizeb knew about the movement of planets in the solar system. It took the form of a drawing and some writing. Each disc representing a planet in Nivizeb' mental model became a circle on the papyrus. This model was independent from any implementation technology. This would be a PIM according to the MDA classification.

Then Nivizeb produced a specification model, called the Water Solarus Papyrus (Figure 12.a). This specification model specified the Water Solarus to be build, a physical system that Nivizeb wanted to use to simulate the movement of the planets. Though it didn't give all the implementation details, this model was a PSM. The Water Solarus was designed as a set of physical pieces representing planets, each pieces floating in a pool in the centre of the patio of the temple, each pieces being connected to the others by strings of reeds constraining the movements of pieces on the surface of the pool. The Water Solarus Papyrus was in fact an Platform Dependent Model which took the form of an annotated version of the Fidus Papyrus. Nivizeb had annotated each model element with an indication of how this element had to be represented. In this annotated model, each planet (which was represented as a disc in Nivizeb mental model, and as a circle on the Fidus papyrus) was decorated by indications of the size, shape and material for the piece to be build. Nivizeb was not very clever with his hands, so he gave this annotated specification model to the workshop of the temple. The workers were warned: the resulting system had to be valid.

The physical system was built successfully from this model. Nivizeb checked the validity of the Water Solarus with respect to the specification model he gave. Nobody was thrown to the crocodiles. Note that, as shown in figure Figure 12.a, the Water Solarus can be considered itself as a model of the solar system, though it is the end product of the engineering process. Nivizeb used the Water Solarus to improve his knowledge about the solar system. He used the Water Solarus as a descriptive model. During the first month, he sometimes found convenient to use the Fidus Papyrus directly. This was possible because all the models produced represented the solar system.

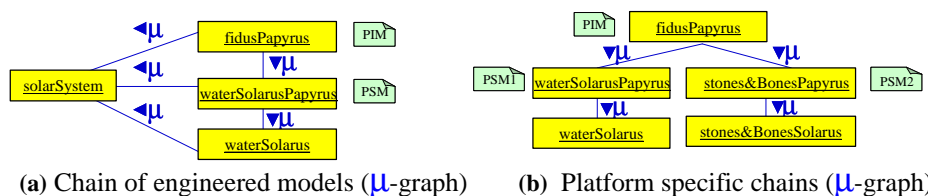


Figure 12 Examples of model engineering

*Nivizeb used the Water Solarus during 7 years. During that time he had made a lot of improvements to it. One day some foreigners, coming from some oriental countries arrived at the temple. They brought with them a new technology. There were excellent craftsmen and had great skills working with stones, bones and leather. Nivizeb decided to migrate his system to this new platform. He drawn frantically a new specification papyrus (Figure 12.b). From the Fidus Papyrus PIM, he derived another PSM: for each planet, he gave an annotation indicating which colour of stones had to be use, how bones had to carved and placed, etc. The new system was successfully implemented by the foreigners, who marked a point against the crocodiles.*

The story above provides an example of successful platform migration. However, some historians are in doubt about the accuracy of this story. Another version of the same event is related later in this paper.

### 5.8 Model (Driven) Forward Engineering

The mega-model introduced in this paper does not attempt to model the engineering process that leads to the production of models. In [35], Karagiannis and Kühn elaborate a richer mega-model defining concepts such as steps, modelling procedures, modelling techniques, modelling methods, etc. Without going into further details, what is clear is that the modelling steps clearly depends on the method and on the engineering discipline. Model engineering is about producing iteratively more and more refined models, and ultimately end up in the production of a final system. The notion of refinement cannot however be defined in general because unfortunately there is not such thing as a "definitive scale of abstractness for classifying models". Once again this really depends on the method and on the engineering discipline. Nevertheless, *forward engineering* can be defined as following:

**"Forward engineering is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system." [36]**

A rigorous forward engineering method will ensure that each step actually transforms a specification model into a valid system. Nivizeb method was not rigorous. He decided at each step if the crocodiles had to be fed up or not, but with no clear criteria. Fortunately, in Software Engineering, model-based specification languages such as Z, VDM or B define mathematically-sound methods that enable stepwise refinement from mathematical models to concrete pieces of code. And at each step the validity of the transformation can be proved. With a lower emphasis on rigor, the MDA approach also sketches a forward engineering process. In that case PIMs are transformed into PSMs, the first models being more "abstract" than the second ones. One of the key idea of the MDE approach is that most of these transformations can be described and (partially) automated thanks to transformation languages. The concept of transformation is a cornerstone of the MDE; it will be described in a further episode of this series.

### 5.9 Model (Driven) Reverse Engineering

In theory, engineering practices should follow the best forward engineering methods available and apply scientific knowledge. As pointed out by Shaw, while this might be

true in some engineering disciplines, this is far from true in the case of software engineering [37]. And that's where everything goes wrong.

While most of the papers in the MDA literature imagine an ideal software world populated by neat models everywhere, the situation is radically different in industry. After 50 years of software development, almost all software companies are still *code-centric*. Despite the advances of specification and modelling languages, the core activity in "software engineering" is still to write code and make it evolve by hand. In spite of their advantages, formal methods have failed to find their path to industry, except in the domain of critical systems. In the last decade, UML received more attention. Nowadays, UML is sometimes used during the analysis or design phase. However, in many cases the specification models produced during these phases remain *contemplative* rather than *productive* [38]. In software industry the code is still the "holder of the truth", the models are not.

By contrast, documenting code afterwards is common practice. Though this usage of UML was not necessarily anticipated, this is not surprising. The huge majority of software systems on the planet have *not* been designed using UML. Even when it was the case, the original UML diagrams typically became obsolete because they were not maintained. In other words, UML is often used to build descriptive models rather than specification models. Though this should not be the case in theory, in practice *reverse engineering* is as important as forward engineering.

As pointed out by Chikofsky and Cross [36], the term "reverse engineering" takes its root in the analysis of hardware systems such as microprocessors, where producing descriptive models from finished systems is a common practice. These authors define reverse engineering as following:

*"Reverse engineering is the process of analysing a subject system, to (1) identify the system's components and their interrelationships, and (2) create representations of the system in another form or at a higher level of abstraction." [36]*

Model Reverse Engineering is just producing descriptive model from existing systems that was previously produced somehow. The last part of this sentence is important, at least to make the distinction between Modelling and Reverse Engineering. In fact Modelling and Reverse Engineering both refer to the activity of creating descriptive models. Replacing "reverse engineering" by "modelling" in Chikofsky and Cross definition would still produce a good definition. However when Nivizeb produced the Fidus Papyrus to model the solar system, he did some *modelling*. When he studied an oriental abacus to build later his own abacus, he did some *reverse engineering*. This is because the abacus under study had previously been engineered somewhere else. Note that Nivizeb had not only reverse engineered foreign systems (this is the dark side of reverse engineering); the next section shows that very often he applied reverse engineering to his own systems.

### **5.10 Engineering = Forward Engineering + Reverse Engineering**

In the 90's reverse engineering was seen as a way to deal with legacy systems. Now, each day it becomes more and more apparent that reverse engineering should be instead closely integrated with forward engineering to support a smooth evolution of software.

For instance in [41], Demeyer, Ducasse and Niestraz show that large and modern systems soon become legacy if not constantly reengineered. This leads to the following equation.

$$\mathbf{Engineering} = \mathbf{Forward\ engineering} + \mathbf{Reverse\ Engineering}$$

This equation has no formal meaning. It just aims at stressing the importance of reverse engineering in engineering disciplines. Small systems built in a scholar way are not of interest in the context of this paper. We are more concerned by the evolution of large scale industrial software products; those software products that evolve over years or decades [31][45]. During that time, those systems must accommodate the evolution of technological spaces and platforms. For instance, in [55] we describe how reverse engineering can support the evolution of a very large component-based software systems built by Dassault Systèmes, one of the largest software company in Europe.

*Nivizeb already knew the importance of reverse engineering. Most historians consider the story related above as an idealized version of what really occurred. It is very unlikely that Nivizeb migrated from the Water Solarus to the Stones&Bones Solarus by just making a new drawing a new papyrus. In fact, one of his young assistants suggested to apply the nice forward engineering principles depicted in Figure 12.b, that is producing another PSM, starting from the same PIM. This naive proposal just made Nivizeb angry, and the crocodiles happy. As pointed out before, Nivizeb had spent 7 years in the dark observing planets. Days after days, he had tuned the Water Solarus by adjusting the size and position of discs and strings. Very often, he had removed and added new mechanisms to improve the correctness of his model. He applied a lot of transformation to it. When in the centre of the pool, he didn't care to maintain the Fidus Papyrus. This PIM rapidly became out of sync. In fact this papyrus was even lost in the sand (where Fido found it later). After years of continuous evolution, the Water Solarus had become a legacy, yet very sophisticated system. The PIM and PSM were totally out of date and of no use. Figure 12.b totally fails to recognize the importance of evolution.*

*To fully recover a descriptive model of the Solarus, Nivizeb spent 77 days in the pool; sizing each disc and string, noting down their positions, reconstructing the behaviour of each mechanisms. Within this period he built a new Platform Specific Model expressed in terms of wood discs, reed strings, etc. Then from this PSM he produced a PIM similar to the Fidus Papyrus, but this recovered model was much more accurate. Nivizeb knew that until this point, providing a new specification for the Stones&Bones Solarus was illusory because too risky and too complex. The optimization, improvement and tuning of the water was a valuable asset. It made no sense to built the Bones&Stones Solarus from scratch.*

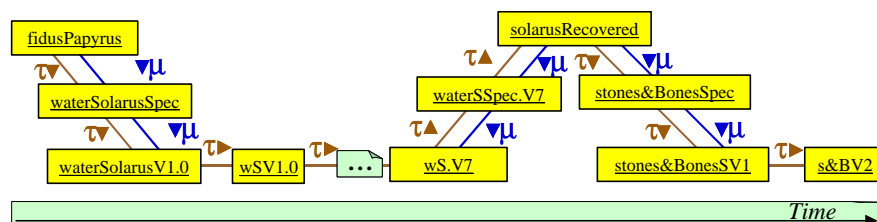


Figure 13 Actual evolution of the Solarus system

The figure above depicts a realistic evolution process. The  $\tau$  links represent successive transformations. This relation will be studied later in this serie.

Decades after decades, Nivizeb observed that evolution process for large real-world systems could be modelled by a sequence of pyramids in the desert. Figure 14 shows a graphic with time on the horizontal axis and  $\mu$  abstraction (🦎) on the vertical axis. In fact, Nivizeb discovered that pyramids could be used to model many real-world phenomena. For instance, the story of Nivizeb' meta-pyramids [29] relates how pyramids can be viewed as models for another abstraction dimension referred as the  $\tau$  abstraction (🦎), the meta-dimension.



Figure 14 Real-world system evolution

### 5.11 Towards the integration of reverse engineering and model engineering

Nivizeb was really a pioneer. At his time, he understood the complementarity of forward and reverse engineering, especially in the context of model engineering. Unfortunately this knowledge disappeared with him. Nowadays, the software engineering field is populated by various communities that exchange too little [33]. In particular, the MDE research community and the reverse engineering community have certainly a lot to share because models are cornerstones of both disciplines. Though more than ten years ago, Chikofsky and Cross established the foundations of reverse engineering by speaking in terms of "views" and "abstractions", their seminal paper [36] could be rewritten by replacing these terms by "descriptive models". The intersection between these two disciplines has received little attention, but this situation is changing thanks to cross-boundaries research papers (e.g. [42][43][44][45]), PhDs (e.g. [47]), research workshops (e.g. [48][50][49]), and industrial workshops (e.g. [52]). In particular, recently the OMG launched the ADM project to merge Reverse Engineering and MDA [51].

### 5.12 Model Engineering and Model-Driven Engineering

To conclude with the engineering of models, let's define two terms. In the literature *model engineering* and *model-driven engineering* are usually used interchangeably, but the adjective driven is meaningful.

*Model engineering is the disciplined and rationalized production of models.*

*Model-driven engineering is a subset of system engineering in which the process heavily relies on the use of models and model engineering.*

For instance when Nivizeb applied Model Engineering to build the Solarus, his goal was ultimately to produce a model. This was not the case however when Nivizeb led the construction of pyramids. He certainly produced a lot of models to drive this huge system engineering project, but this was to control the production of systems. This included models for the architecture of the pyramid, but also models of the resources to be used, of roads and boats to be built, planning models, etc.


Interestingly, the use of the term model-driven is justified in the MDA specification [22]: "*MDA is an approach to system development, which increases the power of models in that work. It is 'model-driven' because it provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification.*". This argument is compatible to the definitions given above.

### 5.12.1 Summary

This section can be summarized by the following statements.

- Model engineering leads to complex nets of  $\mu$  links.
- Models naturally support separations of concerns.
- Various actors with different skills and concerns usually use different models.
- Various technological spaces can be used to manage models.
- There is no such things such as the "best" technological space.
- Bridges between technological spaces is of fundamental importance.
- Some models are more human-oriented than others.
- If a chain of  $\mu$  links is transitive, intermediate models are often omitted.
- Engineering complex systems implies mastering many technological spaces.
- Drawing a map of technological spaces and bridges is an important issue.
- Specification (or prescriptive) models usually specify system to be built.
- A system can be *valid* or *invalid* with respect to a specification model.
- Descriptive models usually describe existing systems.
- A descriptive model can be *correct* or *incorrect* w.r.t. an existing system.
- Forward engineering is refining "abstract" models into more "concrete" models.
- There is no "definitive scale of abstractness" for classifying models.
- The notion of "platform" used in the MDA is not very clear.
- Hence the notion of Platform Independent Model and Platform Specific Model.
- Almost all software companies are still *code-centric*.
- The core activity in "software engineering" is still to write code and maintain it.
- Large industrial software evolve of years or decades.
- Documenting code afterwards is common practice.
- In an evolving world reverse engineering is as important as forward engineering.
- Forward engineering produces specification models.
- Reverse engineering produces description models from engineered systems.
- Modelling produces description models from existing systems.
- Reverse engineering and MDE community share the concept of models.

## 6 Conclusion

The following mega-model has been introduced in this paper to define the notion of model. Many examples have been given to explain how to interpret the single association called RepresentationOf,  $\mu$  or  in Ancient Egyptian.

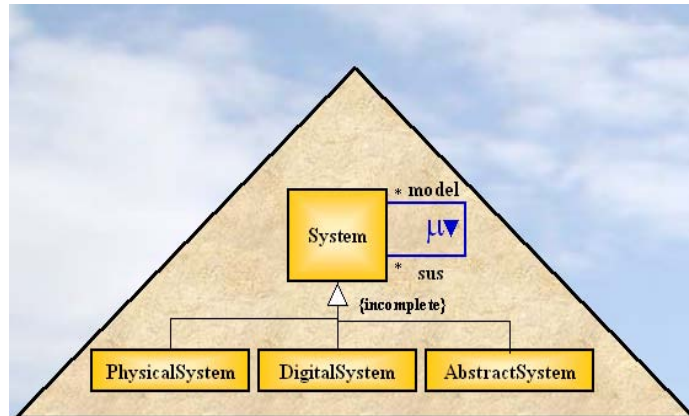


Figure 15 The  $\mu$ -MegaModel

Obviously, this mega-model could be improved substantially. As said before, we deliberately concentrates on a single relation in this first episode. More will be provided in next episodes.

For instance, in this paper all models have been considered in isolation, but another abstraction dimension is necessary to improve model engineering. In episode II [29] the story of Thotus the Baboon will be related. This story will shows how metamodells were discovered in ancient Egypt by Nivizeb with the help of his baboon. This discovery led Nivizeb to design a series of meta-pyramids of different shapes.



Figure 16 Egyptian pyramids built after Nivizeb' meta-pyramids models [29][54]

The first one, the Saqqara "step" pyramid, was build four thousand years ago [54]. At that time, the step pyramid was isolated in the desert. It was the most prominent representation of metamodeling in ancient Egypt. In fact, it played a role similar to the role played today by the MDA meta-pyramid [5]. The MDA set of standards is arranged within a unique pyramidal structure, with a single and unique meta-metamodel on the top, namely the MOF [6][8]. In episode III, it will be shown that seeing the MDA meta-pyramid in isolation was a wrong interpretation of the MDE approach. In this episode the notion of multiple technological spaces [33] has been defined as a fundamental aspect of Model Driven Engineering. This notion will be studied further in episode III.

Though the first pyramids grown separately in the desert, pyramids were later grouped in a structured way as shown in the figure below.



**Figure 17** The Ghiza plateau: a organized set of technological spaces [30][54]

It will be shown in further episodes how concepts such as decomposition, interpretation, syntax, semantics, transformation could fit in the mega-model.

## 7 Acknowledgments

I would like to thanks Jean Bézivin, Jacky Estublier and German Vega, for the fruitful discussions we had on this topic. I would also like to thanks Colin Atkinson and Thomas Kühne to introduce me to Fido and Lassie; Vincent Lestideau to introduce me the Thotus the Baboon, Jean Bézivin to introduce me to Antonio. Thanks to all participants of the Dagstuhl seminar on Language Engineering as well as the participant of the AS MDE project [18]. Discussions largely contribute to this series, but all errors are mine. Finally I would apologize for the errors that might have ocurred during historical narrations.

## 8 Photographic credits

The photos in this paper have been graciously provided by the following individuals or organizations. Special thanks to John Bodsworth for its help.

- J. Bodsworth, The Egypt Archive, <http://www.egyptarchive.co.uk>  
Figure 4.a,b,c,f; Figure 16, Figure 17
- European Space Agency, Figure 4.h
- History Link101, <http://historylink101.net>, Figure 2, courtesy of Kersker.

## 9 References

- [1] Y.F. Chen, "From Ancient Egyptian Language to Future Conceptual Modelling", Conceptual Modelling, LNCS 1565, 1999
- [2] Series "From Ancient Egypt to Model Driven Engineering", resources available at <http://www-adele.imag.fr/mda>
- [3] D. Chandler, "Semiotics: The Basics", Routledge Editor, ISBN 0415265940, 2001
- [4] J.F. Sowa, "Ontology, Metadata and Semiotics", Lecture Notes in AI #1867, Springer-Verlag, 2000
- [5] OMG, "OMG, "Model Driven Architecture (MDA)", ormsc/2001-07-01, July 2001, available at [www.omg.org/mda](http://www.omg.org/mda)
- [6] OMG, MDA Web Site, [www.omg.org/mda](http://www.omg.org/mda)
- [7] S. Cook, "Domain Specific Modelling and Model Driven Architecture", MDA Journal, January 2004, available at [www.bptrends.com](http://www.bptrends.com)

- [8] OMG, "Meta Object Facility (MOF) Specification" Version 1.4, April 2002
- [9] M. Fowler, "UML Distilled", Addison Wesley, 1999
- [10] S.J. Mellor, K. Scott, A. Uhl, D. Weise, "MDA Distilled: Principle of Model Driven Architecture", Addison Wesley, March 2004
- [11] S.J. Mellor, M.J. Balcer, "Executable UML: A Foundation for Model-Driven Architecture", Addison Wesley, May 2002
- [12] J. Bézivin, "Models as First Class Entities", Presentation at Dagstuhl Seminar 4101, 2004
- [13] J. Rho, "A Comparison of ECMA PCTE and ODMG-93", Technical Report
- [14] A. Uhl, "Model Driven Engineering Is Ready for Prime Time", IEEE Software, Sept. 2003
- [15] S. Mellor, L. Rioux, J. Bézivin, "MDA-components: Is there a Need and a Market?", Common Meeting to: OMG MDA WG/UG and the OFTA, June 2003, available at [www.omg.org/docs/ad/03-06-01.pdf](http://www.omg.org/docs/ad/03-06-01.pdf)
- [16] OFTA, "Ingénierie des Modèles - Logiciels et Systèmes", Observatoire Français Des Techniques Avancées, OFTA, ISBN 2-906028-16-9, May 2004
- [17] INRIA, "Model Transformation at INRIA", Institut National de Recherche en Informatique et Automatique, Web site at <http://modelware.inria.fr>
- [18] CNRS, "Action Spécifique CNRS MDA", Centre National de Recherche Scientifique, Web site at <http://www-adele.imag.fr/mda/as>
- [19] S. Gérard, P.A. Muller, "TopModL Initiative", Web site at <http://www.topmodel.org>
- [20] "Joseph Fourier, Hier et Aujourd'hui", Papyrus Thema, Université Joseph Fourier, 2003
- [21] Bibliothèque Nationale de France, "Description de l'Égypte", 1809-1828, Digital version available at [www.bnf.fr](http://www.bnf.fr)
- [22] OMG, "MDA Guide Version 1.0.1", omg/2003-06-01, June 2003
- [23] E. Seidewitz, "What Models Mean", IEEE Software, September 2003
- [24] C. Atkinson, T. Kühne, "Model-Driven Development: A Metamodeling Foundation", IEEE Software, September 2003
- [25] J. Bézivin, "In Search of a Basic Principle for Model-Driven Engineering", Novatica Journal, Special Issue, March-April 2004
- [26] J. Bézivin, O. Gerbé, "Towards a Precise Definition of the OMG/MDA Framework", Proceedings of ASE'01, November 2001
- [27] J. Alvarez, A. Evans, P. Sammut, "MML and the Metamodel Architecture", available from [www.puml.org](http://www.puml.org)
- [28] J.M. Favre, "Metamodel-driven Reverse Engineering - Stories of the Dagktis Stone and of the Rosetta Stone", presentation at Dagstuhl Seminar on Model Driven Approaches for Language Engineering, March 2004, available from <http://www-adele.imag.fr/~jmfavre/>
- [29] J.M. Favre, "Foundations of Meta-Pyramids: Languages and Metamodels - Episode II: Story of Thotus the Baboon", post-proceedings of Dagstzul Seminar on Model Driven Reverse Engineering, May 2004, available at [www-adele.imag.fr/~jmfavre](http://www-adele.imag.fr/~jmfavre)
- [30] J.M. Favre, "Foundations of Metamodel (Driven) (Reverse) Engineering - Episode III: Story of the Ghiza plateau and of the Rosetta Stone", available at <http://www-adele.imag.fr/~jmfavre>
- [31] J.M. Favre, "Meta-models and Models Co-Evolution in the 3D Software Space", Proceedings of ELISA, Workshop on the Evolution of Large scale Industrial Software Applications, Joint workshop with ICSM, Sept.2003, available at [www-adele.imag.fr/~jmfavre](http://www-adele.imag.fr/~jmfavre)
- [32] OMG, "UML2.0 Infrastructure Specification", ptc/03-09-15, September 2003

- [33] I. Kurtev, J. Bézivin, M. Aksit, "Technological Spaces: an Initial Appraisal", CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002
- [34] R. Hausser, "Complexity in Left-Associative Grammar", Theoretical Computer Science, 1992
- [35] D. Karagiannis, H. Kühn, "Metamodelling Platforms", Third International Conference EC-Web – Dexa 2002, LNCS 2455, September 2002
- [36] E.J. Chikofsky, J.H. Cross, "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software, January 1990
- [37] M. Shaw, "Prospects for an Engineering Discipline of Software", IEEE Software, 1990
- [38] J. Bézivin, "MDA: From Hype to Hope, and Reality", Guest talk at UML'2003
- [39] D. Exertier, O. Kaht, B. Langlois, "PIMs Definition and Description to Model a Domain", MASTER Project, Model-driven Architecture inSTRumentation, Master-2002-D.2.1-V1.0-public, December 2002
- [40] P. Klint, R. Lämmel, C. Verhoef, "Towards an engineering discipline for Grammarware", submitted for publication, available at <http://homepages.cwi.nl/~ralf/>
- [41] S. Demeyer, S. Ducasse, O. Nierstrasz, "Object-Oriented Reengineering Patterns", Morgan Kaufman Publishers, 2002.
- [42] J. Bézivin, N. Ploquin, "Tooling the MDA framework: a new software maintenance and evolution scheme proposal", Journal of Object-Oriented Programming, 2001
- [43] L. Bouillon, J. Vanderdonckt, J. Eisenstein, "Model-Based Approaches to Reengineering Web Pages", International Workshop on Task Model and Diagrams for user interface design, TAMODIA 2002
- [44] P. Van Gorp, H. Stenten, T. Mens, and S. Demeyer. "Enabling and using the UML for model driven refactoring", 4th International Workshop on Object-Oriented Reengineering. TR 2003-07, University of Antwerp, 2003
- [45] J.M. Favre, "CacOphoNy: Metamodel-Driven Software Architecture Reconstruction", Working Conference on Reverse Engineering, Nov. 2004
- [46] J.M. Favre, "Towards a Megamodel to Model Software Evolution Through Transformation", SETRA Workshop, ENCTS 2004
- [47] J.M. Sprinkle, "Metamodel-Driven Model Migration", PhD, Univ. of Nashville, Aug. 2003
- [48] J.M. Favre, M. Godfrey, A. Winter Editors, "ateM2003: 1st International Workshop on Metamodels and Schemas for Reverse Engineering", Electronic Notes in Theoretical Computer Science, Vol. 94, at the Working Conference on Reverse Engineering, Nov. 2003
- [49] A. Winter, J.M. Favre, M. Godfrey, "ateM2004: 2nd International Workshop on Metamodels, Schemas and Grammars for Reverse Engineering: Integrating Reverse Engineering and Model Driven Engineering", WCRE 2004
- [50] J. Bézivin *and al.*, "Model Driven Legacy Evolution: Tools and techniques to facilitate development of adaptable enterprise systems", Workshop at EDOC 2004
- [51] OMG, "A Model-driven Approach to Modernizing IT Systems", Workshop, March 2004
- [52] OMG, "Architecture-Driven Modernization (ADM)", <http://www.omg.org/adm/>
- [53] A. Kleppe, S. Warmer, W. Bast, "MDA Explained. The Model Driven Architecture: Practice and Promise", Addison-Wesley, April 2003
- [54] J. Bodsworth, "The Egypt Archive", <http://www.egyptarchive.co.uk/>
- [55] J.M. Favre, *and al.*, "Reverse Engineering a Large Component-based Software Product", European Conf. on Software Maintenance and Reengineering, CSMR'2001