Modeling Issues: A Survival Guide for a Non-expert Modeler

Emilio Rodriguez-Priego, Francisco J. García-Izquierdo, and Ángel Luis Rubio

Departamento de Matemáticas y Computación
Universidad de La Rioja
Edificio Vives, Luis de Ulloa s/n
E-26004 Logroño (La Rioja, Spain)
{emilio.rodriguez,francisco.garcia,arubio}@unirioja.es

Abstract. While developing an integral security model to be used in a Service Oriented Architecture (SOA) context, we find a lot of ambiguities and inaccuracies when authors speak of models, metamodels, profiles and so on. This led us to study a great number of references in a search for precise definitions to help us to address our research. Our study and discussions were so extensive that we are convinced they will be a valuable contribution to the community. In particular, in this paper we present several Reference Concept Maps that depict graphically a large number of definitions with their associated bibliographical references. Nevertheless, we truly believe that there are still a lot of concepts to be clarified and that this clarification is essential so that basic modeling concepts can be best used by non-expert modelers.

Keywords: Modeling concepts, Reference Concept Maps, Metamodeling issues, Stereotypes.

1 Introduction

To be clear from the very beginning: our goal was not to write a paper like this. Our main current research interest concerns security, more specifically the development of an integral security model to be used in Service Oriented Architecture contexts [1]. As a first step of our research we began a deep revision of the literature that quickly showed us that a majority of approaches related to our field of interest define either a UML Profile or a MOF/UML metamodel. But this situation only gave rise to new questions: Which approach is best suited for our needs? How can we evaluate each of them (considering that they even provide different results)? And most important, what are the really essential concepts underlying each approach?

From here we began a nearly endless journey (with a lot of branches) through a vast amount of references dealing with models, profiles, metamodels and with modeling concepts in general. Unfortunately, we came across the fact that there is no apparent consensus about the terminology, not even about what should be the minimal set of basic modeling concepts. In our opinion this is a real obstacle on the way to the development of a specific model/metamodel, especially for people like us, who have

D.C. Petriu, N. Rouquette, Ø. Haugen (Eds.): MODELS 2010, Part II, LNCS 6395, pp. 361–375, 2010.

recently arrived in the modeling world coming from the security world. We devoted a great amount of time trying to understand the usually imprecisely defined concepts, the terminology inconsistencies among several authors, the ambiguities in the official standards, and so on. When we were somewhere in the middle of this journey, we met the paper 'Modeling Modeling' [2], presented in the latest edition of the MODELS conference. Unlike that paper, it was not our aim to make 'a contribution towards a theory of modeling'. However, some sentences throughout that paper (especially in the introduction: 'a lot of people are using models […] without knowing it', 'there is […] neither precise description about what we do when we model, nor rigorous description of the relations among modeling artifacts') encourage us to continue our work. The survey that we present in this paper is a summary of our walk-arounds and private discussions about modeling concepts, and we consider that it can be a valuable contribution for the modeling community.

We have revised a total of 200 references regarding modeling, from 'ancient' references of 1956 to extremely recent (dated 2010) ones[1]. Obviously we are not able to include all these references in the paper because of space reasons. Therefore, we have made a selection of 66 papers which we believe are a significant sample of all the articles reviewed. The paper can be regarded as a well suited guide to find documentation related to recognized problems around modeling. More specifically, this guide is structured as follows. In the following section we deal with those we consider to be the basic modeling concepts. One of the main contributions of this section is the use of a new type of concept map [3], which summarizes the most relevant definitions of these basic concepts found in the literature. Section 3 is devoted to discussing some of the most critical issues about modeling, comparing the contributions of a broad spectrum of authors. In the last Section of the paper we present some conclusions.

2 Basic Concepts in Modeling

The basic modeling concepts have been and are still under discussion in the literature. In order to clarify these terms we did an analysis of the main references using concept maps [3]. These maps, which we named *Reference Concept Maps* (RCM), help to identify the key concepts that appear in each of the term definitions and their relationships. A RCM shows main concepts related to other concepts displayed in different colors depending on their conceptual proximity, within a definition, to the main concepts. Links between the concepts include the numbers of the references where the term definitions are given. The use of RCM to summarize definitions can be considered as an original contribution of this paper. This section aims to enumerate aseptically modeling concepts and their definitions without critically evaluating them. We postpone this discussion to section 3.

2.1 Pure Modeling Concepts

Let's start at the beginning, the concept of *Model*. Surprisingly, or not, there is no generally accepted definition of model. The RCM of Fig. 1 shows that, in a first level,

[1] The complete list can be consulted in Citeulike http://www.citeulike.org/group/13305 or Mendeley http://www.mendeley.com/research-papers/collections/1689661/Model-Theory/

model can be identified as different concepts: representation [4, 5, 6, 7, 8, 9], abstraction [10, 11], statement [12], simplification [7, 13, 14], description [6, 15, 16, 17], specification [6, 15], system [18, 19], entity [14, 19, 20], set [21], replacement [12, 14, 18] or subject [22]. In a second level, model is related to terms such as system [7, 10, 11, 12, 13, 15, 16, 18, 19, 21], reality [6, 8], original [14, 19, 20], software [17] or mental construction [9]. Lack of space prevents us from developing all the definitions completely, but the reader can easily do it following each reference number throughout the RCM links.

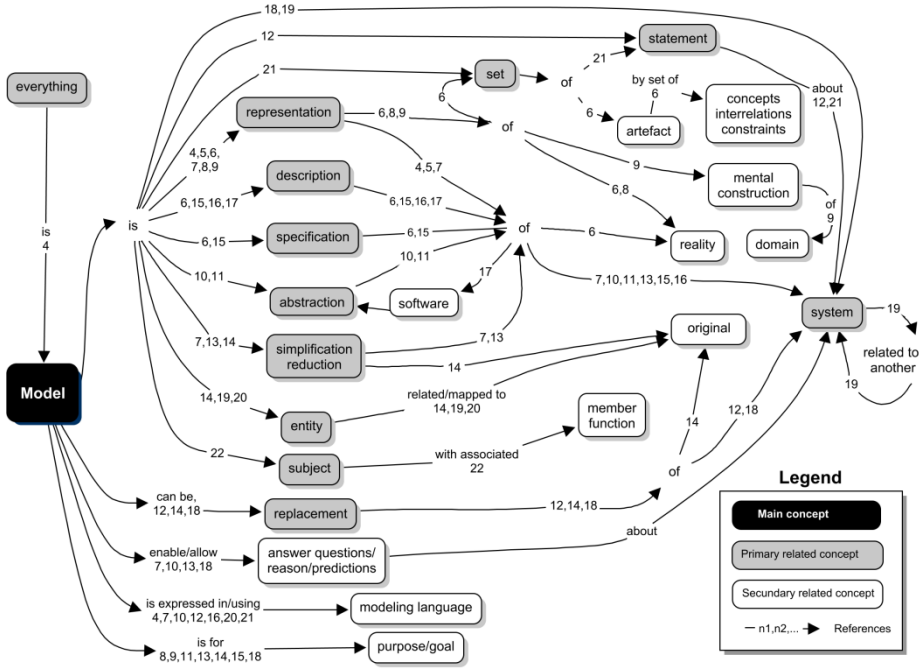

Fig. 1. References Concept Map for model

In the RCM of Fig. 2 we summarize the definitions of other concepts related to model, definitions that seem to confirm the claim in [4] stating that "everything is a model". *Metamodel* is directly related to model [11, 12, 15, 18, 23, 24], specification [4, 6, 21], definition [10, 12, 13, 24], representation [6] and description [7], and indirectly related to concept [7, 13, 24], abstraction [4], modeling [23], or abstract syntax [10, 12, 24]. *Megamodel* is essentially a model of models [5] or a model of MDE concepts [25]. And *Ontology* is generally defined as a type of model or special model [6, 26] or as a model subset [9]. Finally, the reader can also see how the concept of *Modeling Language* is closely related to other concepts apart from the Model concept (Fig. 1). In general, most authors, e.g. [10, 12, 24], identify the *abstract syntax* of a (modeling) language with the concept of metamodel (set of concepts used to create models), and its notation with its *concrete syntax* (textual or graphical representation of those concepts) [27, 28]. Both syntaxes have a many-to-many relation, meaning

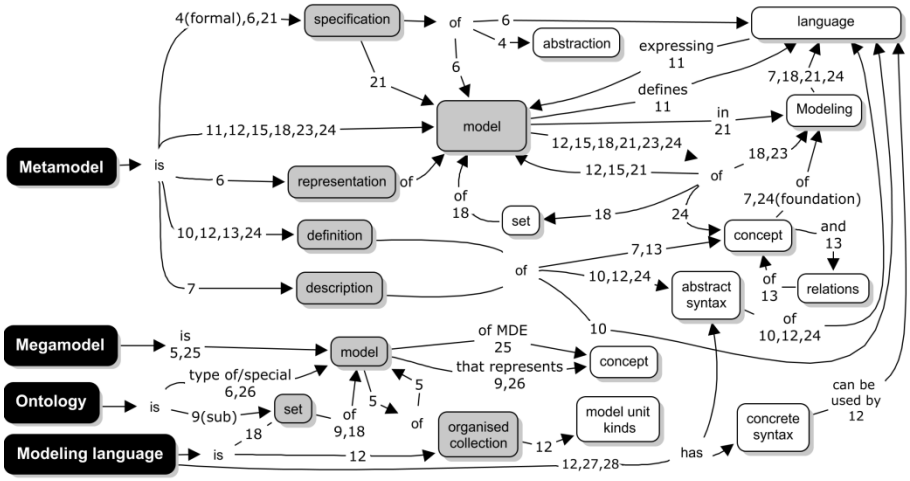

Fig. 2. References Concept Map for metamodel, megamodel, ontology and modeling language

that a model can be expressed in different notations, and that a notation can be used by different languages [12].

2.2 Relationship Related Concepts

The two main types of relationships addressed by literature are *representation/specification* and *conformance*. The former are the basic relations between a model and its modeled entity. Depending on the point of view, the *Representation* relationship, or the models being related by it, have different names: represented-by [4], representation-of [29, 30], descriptive models [4, 21, 31], backward-looking models [12], model as copy [26] or reverse engineering [30]. The same happens to *Specification*, which also has different names: prescriptive models [4], specification models [21], forward-looking models [12], model as original [26] and forward engineering [30]. On the other hand, and regarding to conformance, some authors [4, 7, 29] name the relation between a model and its metamodel *conforms-to*, differentiating it from the previous representation/specification relations used to relate a model and its modeled entity.

2.3 The OMG Approach to Modeling Concepts and Relationships

Classes, Metaclasses and Stereotypes. UML [32] defines *Class* as "a type that has objects as its instances", and type as "a named element that is used as the type for a typed element" (note that this is an 'empty' definition, because the definition contains the defined concept). In addition, UML defines class as "a kind of classifier whose features are attributes and operations". There is no explicit definition of *metaclass* in [32], being understood that a metaclass is a class whose instances are classes. A *stereotype* is "a limited kind of metaclass that cannot be used by itself, but must always be used in conjunction with one of the metaclasses it extends" [32].

Instantiation. OMG names the relationship between an instance and the class to which it belongs *instance-of* [32], and uses it in each one of the different levels of abstraction (instance-class, class-metaclass and metaclass-metametaclass).

Inheritance/Generalization. OMG defines *Generalization* as "a taxonomic relationship between a more general classifier and a more specific classifier" [32].

Package. "A package is a container for types and other packages" [32]. The package concept plays a very relevant role in the modeling based on OMG standards, since "the Model construct is defined as a Package" [33].

Merge. The *Merge* concept is defined by OMG [32] as "a directed relationship between two packages, that indicates that the contents of the two packages are to be combined". The Merge semantics is very similar to that of inheritance, in such a way that it can be considered as the application of inheritance to packages. Nevertheless, it is a complex concept, which requires an extensive description of ten pages [32, pages 161-170], and its application is not without problems [34, 35, 36, 37]. There are simpler proposals for the extension of models that do not use Merge [38].

3 Discussions about Modeling

Some modeling concepts have generated a lot of controversy and endless discussions, whereas other concepts are not clear enough and are still difficult to apply. Next, we list what, in our opinion, are the main causes of confusion in the use of the modeling concepts and theories. They are numbered and classified into three groups: 1) issues about relations between object-oriented concepts and modeling, 2) main points of discussions related to modeling layers, and 3) metamodeling issues, mainly focusing on stereotypes.

3.1 OO Modeling Issues

OOM1–Modeling as an Object Oriented Implementation. The overlapping of object orientation (hereafter OO) concepts (class, instance, object, inheritance…) and modeling concepts (model, representation, specification…) is likely to be one of the main sources of confusion when one tries to understand what the modeling is and how it is applied. The adoption by OMG, responsible for most of the more widely used modeling standards, of an OO approach for the construction of models, has only made the mess bigger. The source of such confusion stems from the fact that both OO concepts and modeling concepts are homomorphic [12]. Concepts such as class, instance, inheritance, etc can be used to unambiguously model *physical* systems. But, when the modeled system is at the same time a model, the use of such terms is confusing and repetitive. Note that then, the same model elements that make up the modeled system are being used to metamodel. This is what several authors call concept replication [39, 40]. Actually, as [25] points out, "object-oriented technologies are used to *implement* MDA standards", instead of being the foundation of modeling. Just as an example, MOF describes its generic term 'Element' as an 'Object' specialization. This may be

very confusing for a reader versed in OO, who cannot easily understand that object and model element are equivalent concepts.

OOM2–Overuse of the Class Concept. Whenever the *class* concept is used in modeling, we have to ask ourselves "class, what class?" Depending on the specification, even on the package being used (e.g. UML-profile), UML and MOF repeatedly employ the class and instance concepts with apparently similar, but essentially different, meanings. This leads to great terminological confusion that forces the modeler to carefully consider the context in which the concept is used, to exactly understand its sense. The specifications even warn about the possibility of the same term being used with different intentions in the same section. E.g., [33] section 18.1.1 says: "Thus, in this clause, when we mention 'Class' *in most cases* we are dealing with the meta-metaclass 'Class' (used to define every meta class in the UML superstructure specification (Activity, Class, State, Use Case, etc.)" (emphasis added). The sentence seems to say that the reader must recognize *in most cases* the actual class to which the section refers. Specifically, as we are later presenting in the stereotypes section, this overuse has caused lively discussions about the true meaning of class in some cases [41, 42].

OOM3–Overuse of the Instance-Of Concept. The fact that, throughout the different UML versions, the term *instance* has been used both to denominate the M0 level instances and the "abstract metaclass" that represents them at M2 level has caused quite a few discussions about the nature of instance [4, 12, 13, 21, 40]. Actually, instance doesn't have a double nature, but, by calling them in the same way, both concepts are being mixed up into a single one. UML version 2 tries to resolve the controversy introducing a new concept, *InstanceEspecification*, defined as "a model element that represents an instance in a modeled system".

Though the term instance has been clarified, the *instance-of* expression is frequently used [10, 32], in such a way that in the OMG approach, virtually every treated concept is instance-of something. Nevertheless, several authors [4, 7, 12, 43] specify that this approach is too restrictive because there are semantic differences among the instantiation undergone between a metamodel and a model, or a model and a modeled real object, or a predicate calculus formula that is generated-by the predicate calculus grammar [7]. E.g., a model is not always an instance of a metamodel [4], though it can be implemented using instances of model elements that, at the same time, are realized using classes. Note that a metamodel could be implemented using relational tables, and its associated models realized using tuples of these tables [6, 44]. If, on the other hand, the Entity-Relationship had been the choice for the modeling, the discussion, and the confusion, would have involved the terms entity-type, entity, association-type, association and model, model element, and so on. The homomorphism existing among these concepts is the root of the confusion. To resolve the situation, some authors propose that, to fully understand the model and metamodel concepts, the discussion should be carried to the ontological level [24].

OOM4–The Object Orientation in Modeling Does not Necessarily Lead to an Object Oriented Model. In UML, the identification of a model element with a class (of M2 level) may lead us to think that object orientation concepts are always present

in modeling. Actually, this is not true, because UML lacks a strict object oriented approach from the modeling point of view. E.g., the static part of a model, which UML names structure, and its dynamic part, named behavior, are usually separated [45]. Though the specification introduces some relationships between both parts of a model, they are not very explicit indeed. Most diagrams focus on a certain single aspect (static or dynamic), but there doesn't exist any diagram that clearly expresses both together, as should be expected from a full object oriented approach. The object orientation appearance of UML is mainly reflected in the class diagrams, in which a class exhibits its collection of attributes (state) and methods. However, the inclusion of methods in class models is paradoxically infrequently used when they are used to express models. Some authors [46] conclude that, even when they are building high level models, they don't need to include methods in their models. In practice, the definition of methods in UML classes are often referred to the management of the class state (set- and get- methods), rather than to the representation of the class behavior.

Summarizing, we could say that whereas UML is implemented using an OO approach, models created using UML rarely follow that approach, except when a class view is used. Moreover in the M2 level (UML specification), the behavioral part of the UML metamodel is devoted to specify the behavior of an hypothetic UML modeling tool.

OOM5–A Model Shouldn't Be More Complicated than its Modeled Entity. As we have pointed out in section 2.1 (RCM of Fig. 1), a model can be considered as a simplification of reality that can be used to handle it in a simpler way. The UML metamodel is an example in which this rule is violated. The UML metamodel is so complex that it can only be analyzed, managed and verified effectively with the aid of tools. Such complexity implies that determining the semantics of the modeling concepts defined in UML is not a simple task if it is done manually, because "the required information is scattered across the metamodel" [47].

The more complex a system is, the greater the possibility of making errors. Some authors have used different formal techniques, which are usually supported by automated tools, to study certain aspects of UML. Thanks to them, a number of errors and inconsistencies across different versions of the UML metamodel have been found, e.g.: [37] on package merge, [48] on associations, [49, 50] on general UML semantics, [51] on generalization and overriding, and [52] on UML 1.x.

3.2 Model Layers Issues

ML1–Lack of a Minimal Top Level Model. UML and MOF are based on a hierarchy made up of four levels named M3 (meta-metamodel), M2 (metamodel), M1 (model) and M0 (run-time instances). This structure is closely inspired by the ANSI IRDS and by EIA/CDIF [53]. The application of these levels in UML is quite confusing. On the one hand [32] states that MOF is the meta-metamodel for UML and other metamodels, saying also that a part of UML called Core is used to specify MOF. The obvious question is then: would Core be the meta-metamodel of both? To solve the problem, the authors name that Core subset of UML *Metalanguage kernel*, a claim that seems to imply that that metalanguage is a constituent part of the metamodel,

contributing even more to the confusion of both terms. The confusion is greater due to the same concepts (package, class…) being extensively used in the definitions of metamodel and metalanguage.

Again, the (over)use of the term *instance-of* to relate concepts belonging to two consecutive levels is the root of the problem. The unsuitability of this approach is reflected in the various problems that arise when trying to implement this scheme without relinquishing its OO root. E.g., some proposals to resolve the problem are: to distinguish between two types of instance-of (ontological and linguistic) [54, 55]; the possibility of instantiation across levels (deep instantiation) [40, 56, 46]; to differentiate between linear hierarchies and non-linear hierarchies (the instantiation is also produced within the same level) [43]; to define relationships between levels different from instance-of, e.g., conforms-to, represented-by [4, 12] and others.

The self-model character that both MOF and CDIF give to the M3 level is also subject to analysis. Without leaving the 4-levels approach, [57] proposes a somewhat simpler top self-model alternative (14 classes compared to 18 in Ecore and 24 in MOF). [21] gives the name *minimal reflexive metamodel* to that metamodel in which "any statement in the minimal reflexive metamodel can be represented in terms of elements of the minimal reflexive metamodel". The paper also analyses the problems of its application. [43] discusses the pros and cons between a recursive top level metamodel and an axiomatic top level metamodel. [12] analyzes the problems driven from the fact that MOF is not minimal, concluding that "true self-model should be minimal". To our knowledge, despite the fact that several authors [12, 37] have expressed its need, there is no proposal of minimal top model (recursive or axiomatic) that helps to avoid these problems, clarifies the modeling theory and constitutes the true kernel of modeling. It is understood that this is an open research problem that poses many difficulties. E.g. [12] suggest the use of two ontological universal concepts: "things" (nodes) and "connections between things" (arcs). However, even with this simple approach, the authors find problems, presenting an example in which a certain concept can be considered as a node or as an arc.

ML2–Bidirectionality and Cardinality between Model and Modeled System. A central issue that always comes up in any discussion about the essence of modeling is the use of a model as a *description* or as a *specification* of a modeled system. As we have mentioned before (sec. 2.2), literature uses different names to denominate the description and specification of models. We must take into account that these terms are always used to characterize the way the model is used, not to label the model itself.

When used as a description, the model represents (describes) an existing system, whereas, when used as a specification, the model specifies (prescribes) a system that doesn't exist yet. This bidirectional feature of modeling has great importance for MDE, since it allows understanding and the application of the modeling to the description of existing models or to the generation of new ones. Nevertheless, the bidirectionality is lost when the OMG specifications are used, mainly due to instance-of being the only relation between modeling levels. This means, e.g., that in UML it is easy to create new instances belonging to a system specified by a model, but, in turn, UML does not provide a mechanism to check that a certain entity is described by a

model. A conscientious revision of MOF may show this bidirectionality in certain situations, only when the instances have been previously created ([23] sec. 9.2).

[58, Fig. 3] considers another type of model, called *explorative*, which is at the same time prescriptive and descriptive and in which "modifications are applied to the model rather than to the real system". "When their effect seems to be positive, the modifications are applied to the original". This approach is also addressed by [30], which calls them *models at runtime*. The complexity of this solution lies mainly in two aspects: 1) the model must keep up with the changes in the system and viceversa; and 2) the system needs to gain access to the model. In this scenario, the models are not necessarily represented by a set of objects. Instead, they make up a repository of modeling elements, which constitutes by itself a system that interacts with the systems that it specifies or describes.

The cardinality of the relationship between elements from one level to the next (models and modeled systems) is another important aspect not addressed in UML/MOF. [12] proposes an interesting classification, distinguishing among (1) *isotypical mappings*, "as those that map one model entity to one system under study (SUS)" entity; (2) *prototypical mappings*, "that map one model entity to a set of SUS entities give by example"; and (3) *metatypical mappings*, that "is prototypical mappings given declaratively".

3.3 Metamodeling Issues

MM1–Stereotypes issues. OMG provides a widely used alternative to the construction of metamodels called UML Profiles [59, 60, 61], redefined in the latest revision of the standard [62]. The first problem we encounter when applying this option is to decide whether, instead of using it, it is more convenient to use an approach based directly on MOF metamodeling. The specification authors [32], after describing the mechanism, admit that "there is no simple answer for when you should create a new metamodel and when you instead should create a new profile", deducing that there is an answer to that question, but that the answer is complicated. However, it is necessary to answer that question, since from that claim it is also deduced that the MOF and UML Profiles approaches are so significantly different that an inappropriate choice in an entire application can compromise the result. Next, we discuss the most relevant aspects of the semantics associated with the definition of a UML Profile, just as they are defined in [32, 33], aspects that give an idea of its complexity, as well as the difficulties that its application entails.

MM1.1–An UML-Profile is a package. If we trace the terms involved in the package concept definition throughout the UML specification [32, 33] we find a complex labyrinth of cross-definitions ('packageable element', 'element', 'concept', etc.) scattered across the modeling levels. This fact is a confirmation of the opinion stated in [47] about the understandability of the UML semantics, which can only be addressed using "query/extraction tools". Another example of this complexity is the fact that the UML-Profile specification provides for the possibility of inheritance between profiles, a feature that is rarely used.

MM1.2–In what level is UML-Profile defined? The M2 elements of a UML-Profile are defined using a M3 concept (*Stereotype*), which is materialized in a M2 concept [32, 33]. The problem with this approach is that stereotype is a specialization of another concept, called again 'class', that embraces all UML Elements (e.g., Actor, Activity, etc. of M2 level) and, therefore, it should be defined at the M3 level (actually it should be MOF::Class). To solve the problem, the authors resort to the trick of using the implementation of MOF::Class defined in the UML-InfrastructureLibrary package, establishing that Profiles::Class inherits from InfrastructureLibrary:: Core::Constructs::Class. Though it seems correct, a new problem arises, because, having forced the definition of M3 level concepts at the M2 level, it isn't possible to perform the XMI serialization of a model. To do it, not surprisingly, the authors have to define a correspondence between Stereotype and MOF:Class and between Profile::Class and MOF::Class [33] section 13.1.6]. This non-trivial usage of an M3 concept by an element of M2 has led to many discussions and criticism of the semantics of UML-Profile (see e.g. [41, 42]).

MM1.3–The extension association. [33] introduces another type of association named *Extension*, which links a Stereotype (M3) with a Profile::Class (M3). As expected, Extension is a specialization of InfrastructureLibrary::Core::Constructs::Association. Despite a stereotype being a Profile::Class specialization, to extend a stereotype with another stereotype is not allowed, whereas, though with some restrictions, it is allowed for a stereotype to participate in associations ([32], sec. 13.1.6). Note that this allows the definition of a stereotype that, for example, extends to both Actor and State concepts, and that, besides, is associated with another stereotype that extends to an activity. The specification [33], sec. 18.1.2] leaves in the modeler's hands that "the specialized semantics do not contradict the semantics of the reference metamodel".

MM1.4–Interoperability, not yet. One of the main arguments for using UML Profiles is that they are supported by various UML editors, which facilitates the exchange between the different tools. Nevertheless, this theoretical interoperability is not always feasible due to: (1) there are several mutually incompatible versions of XMI and UML; (2) the XMI standard does not consider the diagram representation, and, currently, this functionality is covered by proprietary extensions; (3) usually tools do not implement the whole standard because of the complexity of the OMG specifications. In 2006, OMG issued the UMLDI [63] standard, with the purpose of supporting "storage and exchange of information pertaining to the layout of UML Models". In practice, as admitted by OMG, "the DI standard itself is not precise enough to enable consistent exchange of diagrams between different tools" [64, pag. 21], and it is currently subject to a revision process that finally will enable its adoption by the UML tool providers.

MM2–Inheritance and Metamodeling. Two types of relationships are the basis of object orientation: *instance-of*, between an instance and its class, and *inherits-from* between a subclass and its superclass [4]. Although both relationships are applied to different elements, some confusion may appear when they are described using the natural language. E.g., the integer '8' can be obtained from the class 'Integer' and, therefore, we can say that '8 is an integer'. Similarly, if 'Animal' is the superclass of

'Cat', we can say that 'cat is an animal'. In both cases, 'is-a' has different meanings. In the first case, 8 is an instance of the class Integer, whereas an instance of Cat is also an instance of Animal (thanks to the transitivity, it is an instance of Animal *because* it is an instance of Cat). Papers such as [65, 66] analyze the consequences derived from the use of the inheritance or the *instantiation across levels* in modeling (metamodeling). [4] studies the use of the inheritance and the instantiation relationships in OO, comparing them to the conforms-to and represented-by relationships in the modeling theory. On the other hand, [6] defines a similarity relationship that embraces different relationships: subset-of, is-described-by, is-represented-by, instance-of and is-a. The paper concludes that 'is-a' is a specialization of subset-of and is-described-by, but not of instance-of, and that described-by and instance-of are specializations of is-represented-by.

The confusion that, from a generic point of view, exists between *classification* (a concept close to instantiation) and *generalization* (inheritance) has been recently analyzed again by [67], highlighting that there still exists a conceptual problem concerning inheritance and instantiation, concepts that are still incorrectly applied. This problem arises more frequently when UML is used, especially due to two reasons: 1) the semantics of the inheritance in UML "is highly complex, scattered over many diagrams, constraints and additional operations sections and is very difficult to understand" [51]; and 2) the confusing definition of stereotypes, placed between the M2 and M3 levels, has resulted in misuses of stereotypes in M1, when the correct modeling technique should have been to use inheritance (see e.g. [42, 68]).

4 Conclusions

The main conclusion that we can draw is that, despite of the many discussions (that have been widely documented in literature), there are still a lot of confusing notions in the field of model-driven engineering. Even after our journey we feel close to the authors of [2], when quoting Ludewig [58], who in turn claims that "nobody can just define what a model is […] endless discussions have proven that there is no common understanding consistent of models". We believe that a solid foundation for a minimum set of modeling concepts is critical to the success of MDE.

Here we have made a modest contribution in this sense summarizing a series of issues that we, mere model users, consider should be resolved. In addition, we propose several Reference Concept Maps, which can serve as a tool from which to continue the discussions. Much work remains to be done by model/metamodel engineers to enable 'classical' software engineers to use model engineering concepts and tools in an easy and reliable way. This fact can be verified very graphically since, at the time of writing, on the download page for the formal specification of UML the following discouraging note is presented: "Version 2.0 does not have XML or XSD associated files due to structural problems with the UML metamodel".

Acknowledgments. Partially supported by Comunidad Autónoma de La Rioja, project FOMENTA 2008/01, and by Ministerio de Ciencia e Innovación de España, project TIN2009-13584.

References

1. Rodriguez-Priego, E., Garcia-Izquierdo, F.J.: Securing code in services oriented architecture. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 550–555. Springer, Heidelberg (2007)
2. Muller, P.A., Fondement, F., Baudry, B.: Modeling Modeling. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 2–16. Springer, Heidelberg (2009)
3. Novak, J.D., Cañas, A.J.: The theory underlying concept maps and how to construct and use them. Technical report, Florida Institute for Human and Machine Cognition (2008)
4. Bézivin, J.: On the unification power of models. Software and Systems Modeling 4(2), 171–188 (2005)
5. Barbero, M., Jouault, F., Bézivin, J.: Model Driven Management of Complex Systems: Implementing the Macroscope's Vision. In: ECBS 2008, IEEE Int. Conf. on the Engineering of Computer-Based Systems, Washington, DC, USA, pp. 277–286. IEEE Computer Society, Los Alamitos (2008)
6. Aßmann, U., Zschaler, S., Wagner, G.: Ontologies, Meta-models, and the Model-Driven Paradigm. In: Ontologies for Software Engineering and Software Technology, pp. 249–273. Springer, Heidelberg (2006)
7. Ober, I., Prinz, A.: What do we need metamodels for? In: 4th Nordic Workshop on UML and Software Modelling, pp. 8–28 (2006)
8. Pidd, M.: Tools for Thinking: Modelling in Management Science, 3rd edn. John Wiley and Sons, Chichester (February 2009)
9. Sánchez, D.M., Cavero, J.M., Marcos, E.: On models and ontologies. In: 1st Int. Workshop on Philosophical Foundations of Information Systems Engineering, PHISE 2005 (2005)
10. Kühne, T.: Matters of (meta-) modeling. Software and Systems Modeling 5(4), 369–385 (2006)
11. OMG: MetaObject Facility (MOF) 1.4. Technical report (April 2002)
12. Gonzalez-Perez, C., Henderson-Sellers, B.: Modelling software development methodologies: A conceptual foundation. Journal of Systems and Software 80(11), 1778–1796 (2007)
13. Bézivin, J.: Towards a precise definition of the OMG/MDA framework. In: Proc. of the 16th Int. Conf. on Automated Software Engineering (ASE), pp. 273–280. IEEE Computer Society, Los Alamitos (2001)
14. Stachowiak, H.: Allgemeine Modelltheorie. Springer, Wien (1973)
15. OMG: MDA Guide Version 1.0.1. Technical report (June 2003)
16. Kleppe, A., Warmer, J., Bast, W.: MDA Explained. The Model Driven Architecture:Practice and Promise. Addison-Wesley, Reading (May 2003)
17. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press, Cambridge (April 2006)
18. Favre, J.M.: Foundations of meta-pyramids: Languages vs. metamodels Episode II: Story of thotus the baboon. In: Language Engineering for Model-Driven Software Development, vol. 4101 (2005)
19. Klir, G.J.: Facets of Systems Science. Kluwer Academic Publishers, Dordrecht (August 2001)
20. Asikainen, T., Männistö, T.: Nivel: a metamodelling language with a formal semantics. Software and Systems Modeling 8(4), 521–549 (2009)
21. Seidewitz, E.: What Models Mean. IEEE Software 20(5), 26–32 (2003)

22. Rensink, A.: Subjects, Models, Languages, Transformations. In: Language Engineering for Model-Driven Software Development, Dagstuhl. Dagstuhl Seminar Proceedings, Schloss Dagstuhl, Germany, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), vol. 04101 (2005)
23. OMG: MetaObject Facility (MOF) 2.0 Core specification. Tech. Rep. (January 2006)
24. Kurtev, I.: Metamodels: Definitions of Structures or Ontological Commitments. In: Workshop on TOWERS of models. Collocated with TOOLS Europe (2007)
25. Favre, J.M.: Towards a basic theory to model model driven engineering. In: Proc. of the Workshop on Software Model Engineering (WISME 2004), Joint Event with UML 2004 (October 2004)
26. Sánchez, D.M., Cavero, J.M., Marcos, E.: The concepts of model in information systems engineering: a proposal for an ontology of models. The Knowledge Engineering Review 24(Special Issue 01), 5–21 (2009)
27. Kühne, T.: What is a Model? In: Dagstuhl Seminar. Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2005)
28. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. ACM Trans. Model. Comput. Simul. 12(4), 290–321 (2002)
29. Favre, J.M., Nguyen, T.: Towards a megamodel to model software evolution through transformations. In: SETRA Workshop. ENCTS, vol. 127, pp. 59–74. Elsevier, Amsterdam (2004)
30. Jouault, F., Bézivin, J., Barbero, M.: Towards an advanced model-driven engineering toolbox. Innovations in Systems and Software Engineering 5(1), 5–12 (2009)
31. Favre, J.M.: Foundations of Model (driven) (Reverse) Engineering - Episode I: Story of the Fidus Papyrus and the Solarus. In: Dagsthul Seminar on Language Engineering for Model- Driven Software Development (2004)
32. OMG: UML Infrastructure v2.2. Technical report (February 2009)
33. OMG: UML Superstructure v2.2. Technical report (February 2009)
34. Zito, A., Diskin, Z., Dingel, J.: Package Merge in UML 2: Practice vs. Theory? Model Driven Engineering Languages and Systems, 185–199 (2006)
35. Zito, A., Dingel, J.: Modeling UML2 package merge with Alloy. In: First Alloy Workshop (2006)
36. Bottoni, P., D'Antonio, F., Missikoff, M.: Towards a Unified View of Model Mapping and Transformation. In: Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI-INTEROP), co-located with CAiSE 2006 Conf. (June 2006)
37. Dingel, J., Diskin, Z., Zito, A.: Understanding and improving UML package merge. Software and Systems Modeling 7(4), 443–467 (2008)
38. Barbero, M., Jouault, F., Gray, J., Bézivin, J.: A Practical Approach to Model Extension. MDA-Foundations and Applications, 32–42 (2007)
39. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software and Systems Modeling 7(3), 345–359 (2008)
40. Atkinson, C., Kühne, T.: The Essence of Multilevel Metamodeling. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 19–33. Springer, Heidelberg (2001)
41. Weisemöller, I., Schürr, A.: A Comparison of Standard Compliant Ways to Define Domain Specific Languages. In: Giese, H. (ed.) MODELS 2008. LNCS, vol. 5002, pp. 47–58. Springer, Heidelberg (2008)

42. Henderson-Sellers, B., Gonzalez-Perez, C.: Uses and abuses of the stereotype mechanism in uml 1.x and 2.0. In: Model Driven Engineering Languages and Systems, pp. 16–26 (2006)
43. Gitzel, R., Hildenbrand, T.: A Taxonomy of metamodel hierarchies. Technical report, Department of Information Systems. University of Mannheim (2005)
44. Thomas, D.: MDA: Revenge of the modelers or UML utopia? IEE Software 21(3), 15–17 (2004)
45. Merunka, V.: Critical Assessment of the Role of UML for Information System Development. In: Systems Integration, pp. 445–452 (2003)
46. Gitzel, R., Ott, I., Schader, M.: Ontological Extension to the MOF Metamodel as a Basis for Code Generation. The Computer Journal 50(1), 93–115 (2007)
47. France, R.B., Ghosh, S., Dinh Trong, T., Solberg, A.: Model-Driven Development Using UML2.0: Promises and Pitfalls. Computer 39(2), 59–66 (2006)
48. Milicev, D.: On the Semantics of Associations and Association Ends in UML. IEEE Transactions on Software Engineering 33(4), 238–251 (2007)
49. Shan, L., Zhu, H.: A Formal Descriptive Semantics of UML. In: Liu, S., Maibaum, T., Araki, K. (eds.) ICFEM 2008. LNCS, vol. 5256, pp. 375–396. Springer, Heidelberg (2008)
50. Akehurst, D.H., Howells, W.G.J., Bordbar, B., Mcdonald-Maier, K.D.: Maths vs (Meta) Modelling: Are we reinventing the Wheel? In: ICSOFT 2008, Porto, Portugal (2008)
51. Buttner, F., Gogolla, M.: On generalization and overriding in UML 2.0. In: Proc. UML 2004 Workshop OCL and Model Driven Engineering, pp. 69–83 (2004)
52. Fuentes, J.M., Quintana, V., Llorens, J., Genova, G., Prieto Diaz, R.: Errors in the UML metamodel? SIGSOFT Software Engineering Notes 28(6), 3 (2003)
53. Flatscher, R.G.: Metamodeling in EIA/CDIF—meta-metamodel and metamodels. ACM Transactions on Modeling and Computer Simulation (TOMACS) 12(4), 322–342 (2002)
54. Atkinson, C., Kühne, T.: Model-driven development: a metamodeling foundation. IEEE Software 20(5), 36–41 (2003)
55. Gaševic, D., Kaviani, N., Hatala, M.: Ón Metamodeling in Megamodels. Model Driven Engineering Languages and Systems, 91–105 (2007)
56. Varró, D., Pataricza, A.: A Unifying Semantic Framework for Multilevel Metamodeling. Tech. rep., Budapest Univ. of Technology and Economics (2001)
57. Jouault, F., Bézivin, J.: KM3:A DSL for Metamodel Specification. In: Gorrieri, R., Wehrheim, H. (eds.) FMOODS 2006. LNCS, vol. 4037, pp. 171–185. Springer, Heidelberg (2006)
58. Ludewig, J.: Models in software engineering - an introduction. Software and Systems Modeling 2(1), 5–14 (2003)
59. Bruni, R., Hölzl, M., Koch, N., Lluch Lafuente, A., Mayer, P., Montanari, U., Schroeder, A., Wirsing, M.: A service-oriented UML profile with formal support. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 455–469. Springer, Heidelberg (2009)
60. Rodriguez, A., Fernandez-Medina, E., Piattini, M.: Security requirement with a uml 2.0 profile. In: Proceedings of the First Int. Conf. on Availability, Reliability and Security (ARES), pp. 670–677. IEEE Computer Society, Los Alamitos (2006)
61. Houmb, S.H., Den Braber, F., Lund, M.S., Stølen, K., Informatics, S.T.: Towards a UML profile for model-based risk assessment. In: Critical Systems Development with UML-Proceedings of the UML 2002 Workshop, pp. 79–91 (2002)
62. Selic, B.: What's new in UML 2.0. IBM rational software (2005)

63. OMG: Diagram Interchange Specification, v1.0. Tech. rep. (2006)
64. OMG: Diagram Definition RFP-OMG Document 07-09-02. Tech. rep. (2007)
65. Atkinson, C., Henderson-Sellers, B., Kühne, T.: To Meta or Not to Meta. That Is the Question. Journal of Object-Oriented Programming 13(8), 32–35 (2000)
66. Atkinson, C., Kühne, T.: Profiles in a strict metamodeling framework. Science of Computer Programming 44(1), 5–22 (2002)
67. Kuhne, T.: Contrasting Classification with Generalisation. In: Proceedings of the Sixth Asia-Pacific Conf. on Conceptual Modelling, New Zealand (2009)
68. Atkinson, C., Kühne, T., Sellers, B.H.: Systematic stereotype usage. Software and Systems Modeling 2(3), 153–163 (2003)