Formalization of the UML Metamodel: An Approach Based Upon the Four-Layer Metamodeling Architecture

Marie-Noëlle TERRASSE
Guest Researcher at the National Institute of Standards and Technology (Maryland, USA)
Sabbatical Leave from the LE2I Laboratory, University of Burgundy (France)
and
Marinette SAVONNET
LE2I Laboratory, University of Burgundy (France)

Abstract

Modeling tools move towards increasingly abstract approaches such as metamodels and the four-layer metamodeling architecture. We address modeling tools using the UML metamodel as a core. We propose our four-layer metamodeling architecture based upon a two-fold structure of meta-metamodel and metamodel layers. The meta-metamodel layer is populated with a lattice of modeling paradigms that are instantiated –at metamodel level– as specializations of the UML metamodel. The inheritance hierarchy of metamodels is a mirror of the lattice of modeling paradigms. Different pre-conditions may be used to guard the instantiation of ambiguous modeling paradigms. Non-strict pre-conditions allow a “delayed formalization” of the UML metamodel.

1 Introduction

Under pressure of the increasing complexity of systems being modeled, modeling tools move towards more and more abstract methodologies. One axis of abstraction is based upon the principles of separation and combination of concerns [3, 5, 15] and leads to the development of multi-view models. Each view focuses
on a particular aspect of the system without taking into account any other aspect. As depicted in Figure 1.a, such a multi-view model is defined and controlled by a metamodel which (a) determines the different views and the best representation for each view, (b) specifies—as inter-view constraints—the semantical relationships between views, and (c) provides a global view of the whole system. For example, the UML metamodel proposes 9 complementary views, expressed as diagrams, for modeling of a system. The second axis of abstraction is the metamodeling architecture in which each layer presents an abstract view of its lower layer: instances are abstracted by a model, models are abstracted by a metamodel, metamodels are abstracted by a meta-metamodel. Figure 1.b presents the OMG’s four-layer architecture in which UML is used as a descriptive language on the uppermost three layers.

Along both axes of abstraction, the metamodel plays a major role and many modeling tools rely on the UML metamodel. At the same time, many authors point out ambiguities of the semantics of UML’s metamodel as a problem that must be addressed. For improvement of UML’s semantics two strategies can be used:

- The restriction strategy [1, 6, 7, 10, 12, 13] aims at providing an unambiguous semantics to a restricted part of UML. As presented by Evans & al. [2, 11], the main idea is to strengthen, extend or simplify the metamodel in order to assure the completeness of each ambiguous element.

- The extension strategy is based upon the idea that UML is not one modeling language but a family of modeling languages. Thus, it is necessary to extend UML with a mechanism for specifying which member of the family is to be used. The axiomatic meta-metamodels of Nissen [16], the modeling paradigms of Nordstrom & al. [17], the profiles of the OMG [18], and the prefaces of Cook & al. [8] are such mechanisms. Note that these proposals describe tools that operate at different levels (meta-metamodel or metamodel) of the metamodeling architecture.

Our proposal belongs to the extension strategy. We use a modeling paradigm (similar to those defined by Nordstrom & al) to define a member of the UML family of languages. We structure modeling paradigms into a lattice (similar to the Cook & al.’s inheritance hierarchy of prefaces). We instantiate modeling paradigms into specializations of the UML metamodel: the inheritance hierarchy of specialized metamodels being a “mirror” of the lattice of modeling paradigms. Thus, we base our four-layer metamodeling architecture upon a two-fold structure of modeling
paradigms and metamodels. The formalization of metamodels within the inheritance hierarchy may be controlled by more or less strict pre-conditions guarding instantiation of ambiguous modeling paradigms into metamodels. We say that the formalization of metamodels is “delayed” when the pre-condition is not strict.

Sections 2 and 3 present the meta-metamodel and metamodel layers, respectively, of our metamodeling architecture. Section 4 addresses formalization issues. Section 5 concludes by giving an overview of the on-going work.

2 Meta-metamodel layer

The main feature of our four-layer metamodeling architecture is a two-fold structure of meta-metamodel- and metamodel-layers. In order to justify this structure, we first discuss the use of modeling paradigms at meta-metamodel level, then we explain how they are structured into a lattice, and finally, we address the potential flatness of this lattice.

Modeling paradigms are abstract descriptions of the set of requirements under which systems are being modeled. They can use both natural and formal languages (English, logic, set theory, etc.). Even though this definition seems to place modeling paradigms at metamodel level –in the sense that they are application domain related– we have argued [24] that modeling paradigms are more abstract than metamodels: modeling paradigms are intensional descriptions of the set of possible models and cannot be instantiated directly into models. Thus, in
our architecture, modeling paradigms are assigned to the meta-metamodel layer and are supposed to be instantiated –at metamodel level– as metamodels. Since we are interested in a UML-based metamodeling architecture, we restrict our architecture to the set $Restri_{MP}$ of modeling paradigms that can be instantiated as specializations of the UML metamodel. For example, a modeling paradigm using any non-boolean logic is not relevant to our architecture.

Modeling paradigms are expressed as constraints: for example, specification of particular meanings of inheritance [9, 14], extension of aggregation [21] as well as restrictions on the multiplicity of association relationships. We use the subsumption of constraints, and we define a partial order on modeling paradigms, denoted by $\preceq$, as follows: $mp \preceq mp'$ if and only if the condition of $mp'$ is less restrictive than the condition of $mp$.

If we were working with general constraints, such a partial order would have a bad coverage that results in a flat structure. On $Restri_{MP}$, we can improve the structure induced by $\preceq$ because our constraints are limited to the potential variations between members of the UML family of languages. Thus, we can argue that:

- A generic modeling paradigm, denoted by $gmp$ and corresponding to the UML itself –without any restriction–, is included into $Restri_{MP}$ and its condition subsumes the condition of any other modeling paradigm of $Restri_{MP}$. Thus, the partial order $\preceq$ defines a lattice on $Restri_{MP}$.

- If we exclude the extension mechanism of UML, conditions are related to
the semantical weaknesses of UML addressed by many papers [1, 4, 7, 10, 12, 19, 21]. All possible topics, as well as the set of possible choices for each topic, are circumscribed. It is thus possible to predict the depth of the lattice.

- Each use of the extension mechanism generates an independent sub-lattice (a sub-lattice related to the generic modeling paradigm only). We observe that, in most cases, the extension mechanism relates to fundamental elements and is used prior to any other choices. Thus, the number of independent sub-lattices will be close to the number of different extensions defined.

Figure 2 depicts the lattice that corresponds to the following modeling paradigms, note that only direct subsumptions appear in the figure: (mp₁) The modeling paradigm mp₁ restricts the association relationship to a finite multiplicity. (mp₂) The modeling paradigm mp₂ defines a special meaning of inheritance. The condition of mp₂ is not comparable to the condition of mp₁. (mp₃) The modeling paradigm mp₃ extends the aggregation relationship. The condition of mp₃ is not comparable to the conditions of mp₁ and mp₂. (mp₄ and mp₅) The modeling paradigms mp₄ and mp₅ are specific restrictions of the association relationship (multiplicities equal to 2 and 3, respectively). The condition of mp₁ subsumes both conditions of mp₄ and mp₅. Thus we have mp₄ ≤ mp₁ and mp₅ ≤ mp₁.

This way, we have organized modeling paradigms into a lattice whose depth and the number of incomparable major sub-lattices are controlled. This structure is the basis for the organization of metamodel layer.

3 Metamodel layer

The objective is to build the metamodel layer as a mirror of the lattice of modeling paradigms. By construction, the generic modeling paradigm gmp is instantiated into the UML metamodel. A modeling paradigm is instantiated into a specialization of the UML metamodel by using the tailoring mechanism of UML (constraints, stereotypes and tagged values [20]). Let us denote by \(\mathcal{E}\) the instantiation relationship between modeling paradigms and metamodels. \(\mathcal{E}\) must preserve –through the inheritance hierarchy– the structure of lattice on Restrict\(_{MP}\). Thus, we add the following rule: if a modeling paradigm mp subsumes a modeling paradigm mp′, then the instantiation \(\mathcal{E}(mp')\) inherits from the instantiation
Our major concern is the type of inheritance that will be used, i.e., simple or multiple inheritance. On one hand, it may be interesting to use multiple inheritance because the lattice of modeling paradigms may include convergent ordering links. As depicted in Figure 3.a, a modeling paradigm $mp_6$ with two extensions for aggregation and inheritance is subsumed by both $mp_2$ and $mp_3$. On the other hand, we can restrict ourselves to simple inheritance, duplicate the specializations in case of conflict, and complement the lattice when two metamodels may be considered as “equivalent” [23]. In the previous example, depicted in Figure 3.b, modeling paradigms $mp_2$ and $mp_3$ are extended by aggregation and inheritance into modeling paradigms $mp_6$ and $mp_7$, respectively. In fact, we observe that, in most cases, modeling paradigms are defined by referring to only one of the existing paradigms. Thus, we work with simple inheritance and we provide tools to complement the lattice.

As a result, we have built the metamodel layer as an inheritance hierarchy of specializations of the UML metamodel. The organization of metamodels into the hierarchy reflects the organization of modeling paradigms into the lattice. Figure 4, in which instantiations are shown as grey thick arrows, illustrates the two-fold structure of meta-metamodel and metamodel-layers.

We are convinced that this two-fold structure of the uppermost layers of the metamodeling architecture may be an efficient basis for determining—at high level of abstraction— formal operations (comparison, evaluation, distance, etc.) on infor-

---

**Figure 3: Multiple or Simple Inheritance Relationships**

$E(mp)$. We can impose this rule because if $mp$ subsumes $mp'$ then the metamodel instantiated from $mp$ is more general than the metamodel instantiated from $mp'$. 

---

6
4 Formalization issue

As discussed in Section 1, the UML metamodel formalization is addressed by many authors. We use our particular metamodeling architecture to propose a new approach. On one hand, we wish to allow –if possible– the user to “delay” the formalization. Our idea is to view some of the ambiguities of the UML’s semantics as strongly related to the general-purpose scope of UML. In this point of view, it is truly important to be able to cope with ambiguity. On the other hand, if we want to use formal tools on metamodels (for comparison of metamodels, integration of metamodels, etc.), we need to formalize them [2]. The problem of formalizing a metamodel instantiated from a modeling paradigm is that –in the general case– modeling paradigms are not formalized and thus may be ambiguous. We cannot guarantee existence of a unique formal instantiated metamodel.

We propose to deal with this problem by adding a pre-condition to the instantiation relationship $\mathcal{E}$. This pre-condition indicates which properties (in terms of ambiguity) are required of a modeling paradigm to be instantiated. In the following, we propose two “extreme” pre-conditions, and then their compromise. In all of them, any unambiguous modeling paradigm may be instantiated.

- **Strictest pre-condition**: We require that only unambiguous modeling paradigms may be instantiated. In this case, we instantiate the modeling paradigm into a unique formal metamodel. We can have many modeling paradigms that are not instantiated. The two-fold structure of metametamodel and metamodel layers is weak in the sense that the metamodel layer is not an accurate mirror of the meta-metamodel layer: ambiguous modeling paradigms are not mirrored. The metamodel layer is completely formalized -except for the UML metamodel itself- and thus formal operations on metamodels have a minimal pre-condition: they apply to any metamodel except for the root of the inheritance hierarchy. Such a structure is being investigated to measure -at high level of abstraction- the evolution of information systems [22].

- **Weakest pre-condition**: We can allow any modeling paradigm to be instantiated. In this case, the instantiated metamodel may be either formal (if the modeling paradigm is unambiguous) or informal (if the modeling paradigm
is ambiguous). This case corresponds to a strong two-fold structure: the metamodel layer is an accurate mirror of the meta-metamodel layer. The single cause of variations between the two layers is the restriction to simple inheritance. The metamodel layer may contain many informal metamodels. Thus, formal operations on metamodels must have strong pre-conditions in terms of the formality of their parameters.

- **Intermediate pre-condition:** The pre-condition for ambiguous modeling paradigms is not “local” but refers to the inheritance hierarchy of metamodels. We require that any sub-hierarchy with a formal root contains only formal metamodels. This requirement implies that the extension mechanism may not be used to create a heir of a formal metamodel. In terms of modeling paradigms, it means that an ambiguous modeling paradigm may be instantiated only if it has no subsuming modeling paradigm instantiated as a formal metamodel. This requirement guarantees that formal operations apply to any sub-hierarchy rooted by a formal metamodel. The two-fold structure of meta-metamodel and metamodel layers should be strong if we can require modelers to first make the extensions, and then make others choices. Such a structure is being investigated in the context of interoperable Geographic Information Systems [23].

Figure 4: The Two-Fold Structure of Modeling Paradigms and Metamodels
Our conclusion is that, by choosing a stronger or weaker pre-condition for the instantiation of modeling paradigms, we can allow modelers to delay the formalization of metamodels to a varying degree. Such a choice influences the cohesion of meta-metamodel and metamodel layers within our two-fold structure as well as the efficiency of formal tools being applied to the inheritance hierarchy.

5 Conclusion

We have proposed a “flexible” approach for formalizing specializations of the UML metamodel by using our four-layer metamodeling architecture. Such an approach is based upon a two-fold structure in which partially ordered modeling paradigms (at meta-metamodel level) describe which language (or which subfamily) of the UML family is used. This organization of modeling paradigms is emulated at the metamodel layer by an inheritance hierarchy of specialization of the UML metamodel: modeling paradigms are instantiated into metamodels. The pre-condition for the instantiation relationship -in terms of unambiguity of the modeling paradigm- may be more or less strict. We are convinced that an intermediate pre-condition, allowing instantiations as informal metamodels but requiring heirs of a formal metamodel to be formal, is a good compromise.

Our on-going work focuses on three different concerns:

- **Metamodeling architecture**: We need to evaluate the consequences of our delayed formalization, i.e., to determine to what extent the existence of informal metamodels may limit the efficiency of a given formal tool or make the inheritance hierarchy less meaningful.

- **Modeling environment**: We need to propose a tool that can help modelers to express their modeling requirements, i.e. to determine which language of the UML family they want to use. This tool should synthesize the proposals for the formalization of the UML metamodel.

- **Modeling process**: We need to evaluate the proposed pre-conditions in different contexts and compare them to each other. The objective should be to determine –if possible– criteria for choosing the best pre-condition for a given context.
Acknowledgements
We thank people at the National Institute of Standards and Technology that have contributed to this work, discussed our proposal, provided useful comments and proposed many improvements: Christopher E. Dabrowski, Leonard J. Gallager, Lisa J. Carnahan, and George Becker.

References


