Operations on metamodels in the context of a UML-based metamodeling architecture

Marie-Noëlle Terrasse, George Becker, Marinette Savonnet, and Eric Leclercq
Laboratoire LE2I, Université de Bourgogne
B.P. 47870, 21078 Dijon Cedex, France
E-mail: \{terrasse,becker,savonnet,leclercq\}@khali.u-bourgogne.fr

Abstract

In the context of information system engineering, we propose a four-layer metamodeling architecture with a comprehensive set of operations on metamodels. Our architecture enables modelers to use a three-step modeling process: first, giving an informal description of the universe of the discourse (in terms of modeling paradigms); then, defining a corresponding UML dialect (in terms of metamodels); and finally –using the chosen dialect– describing a model of an information system. By using specific properties of our metamodeling architecture, we define formal and semantical operations on metamodels, e.g., integration of metamodels. In this paper we focus on a measure of a semantical distance between metamodels.

1 Introduction

Information system engineering is turning into an increasingly complex process [7] which has to take into account complexity of information systems, deployment environments, exploitation environments, and users’ requirements. In order to cope with such demanding requirements, information system engineering has been using sophisticated abstraction mechanisms: abstraction by conceptualization (four-layer metamodeling architectures) and abstraction by projection (multi-view models). These abstraction mechanisms are implemented by metamodeling environments. These environments also provide modelers with formal languages (logics, set theory, Z notation or VDM languages, etc.) and formal methods (model checking, theorem proving, etc.). Nevertheless, many metamodeling environments additionally provide modelers with more intuitive languages, such as natural languages. Many metamodeling environments are based on UML. In such metamodeling environments, the UML metamodel forms the core of both abstraction mechanisms. The UML metamodel is used for defining a set of views and a modeling language which can satisfy the specific needs of a given application domain. For example, Koch & al. (UWE project for hypermedia design [4]) define additional diagrams and UML-constructs for these diagrams¹. In practice, searching for a convenient metamodel could be really difficult: it would be necessary to know all earlier metamodel extensions (in order to decide whether an existing metamodel can be used or whether a new metamodel needs to be defined), and it would be necessary to master the manner in which existing metamodels have been built (in order to be able to extend them properly). Furthermore, when studying modelers’ practice, it appears that modelers do not work directly with metamodel descriptions: information is provided to them in the form of “semi-formal” descriptions (which can be ambiguous but tend to be more readable). We call such a description a modeling paradigm. The modeling process—in practice—is a three-step process: 1) defining a modeling paradigm (by identifying a modeling paradigm which is close to the needed paradigm, and then building a variant of such a modeling paradigm); 2) instantiating the modeling paradigm into a metamodel; and 3) instantiating the metamodel into a model. As a consequence, metamodels and models do not provide comprehensive information about the actual modeling process: all the initial work (in defining modeling paradigms) is lost. Such an “elimination” of meaningful information from a metamodeling architecture can lead to metamodel proliferation since the easier choice for a modeler is to derive his/her own metamodel directly from the UML metamodel (without reusing existing extensions of the UML metamodel). The first effort to avoid

¹Koch & al. propose to model web-systems by navigational space, navigational structure, static and dynamic presentational diagrams. A navigational space diagram describes classes that users are allowed to visit, together with links available for navigation between these classes. A navigational structure diagram describes which types of navigation are available for each link (e.g., guided tour, index). A static presentational diagram describes how objects are presented to users (i.e., interfaces), etc. Koch & al. propose new stereotypes such as \langle navigational class \rangle, and \langle guided tour \rangle which are stereotypes of class.
metamodel proliferation has been made by the OMG which defined specific "profiles" for various application domains [8, 9]. Another approach is to organize existing metamodels into a hierarchy [1, 2]: each extension is a heir of an existing metamodel. In such a case, quality of the hierarchy of metamodels (which must reflect semantical links between metamodels) is a major requirement. Thus, we need to organize metamodels into a structure that makes them easy to reuse by modelers. Our proposal is based on a mirroring descriptive structure: the first part of the description is informal and easily usable by any modeler, the second part of the description is formal and can thus form a basis for application of formal methods. In order to make such a two-fold description meaningful, the two parts of a description must be closely tied together.

2 Overview of our metamodeling architecture

Modeling paradigms describe—in terms of concepts that are interrelated by constraints—the semantics modelers assign to the real world. Our hypothesis is that there is a one-to-one correspondence between modeling paradigms and metamodels. This hypothesis leads to the metamodeling architecture described in the following paragraphs and depicted in Figure 1. We first present, using an example, the two uppermost layers of our metamodeling architecture (i.e., a poset of modeling paradigms and an inheritance hierarchy of metamodels). Then we give more details on the mirroring structure and its features: subsumption of modeling paradigms into metamodels, and partitioning of UML-constructs and OCL-constraints for a given metamodel.

A poset of modeling paradigms Modeling paradigm descriptions possibly mix several different languages: the English language, logics, the set theory, the Z notation, etc. Modeling paradigms may use a various number of concepts, e.g., class and abstract class, each of them being described with more or less precision. A modeling paradigm \( m_p \) is described by two sets, \( E^p(m_p) \) and \( C^p(m_p) \). The set \( E^p(m_p) \) contains descriptions of elementary concepts, while the set \( C^p(m_p) \) contains constraints between concepts of \( E^p(m_p) \).

We define a partial order between modeling paradigms by using a subsumption relation and we obtain a poset (i.e., partially ordered set) of modeling paradigms.

We denote by \( gmp \) the generic modeling paradigm which corresponds to the standard UML semantics. We restrict ourselves to a set of modeling paradigms that are subsumed by \( gmp \); we denote this set by \( Restrict_{MP} \).

In Figure 1 we depict several examples modeling paradigms of \( Restrict_{MP} \). Modeling paradigms \( m_{p2} \) and \( m_{p4} \) provide time descriptions. \( m_p4 \) which has an additional constraint (i.e., time model uniqueness) is subsumed by \( m_{p2} \) which supports several time models. Modeling paradigm \( m_{p3} \) supports the C2 Architecture Description Language. There is no subsumption between \( m_{p2} \) and \( m_{p3} \).

A mirroring inheritance hierarchy of metamodels Our objective is to build the metamodel layer of our architecture as a mirror of the poset of modeling paradigms: the generic modeling paradigm \( gmp \) is instantiated into the UML metamodel itself (which we denote by \( mm_{UML} \)), and all other modeling paradigms are instantiated into specializations of the UML metamodel (by using UML's extension mechanisms: constraints, tag-values, and stereotypes). Furthermore, we require that each metamodel instantiates some modeling paradigm, and that each inheritance link between metamodels instantiates a subsumption link between modeling paradigms. For example, in Figure 1, the general modeling paradigm \( gmp \) is instantiated into the UML metamodel \( mm_{UML} \) and \( m_{p3} \) is instantiated into a metamodel \( mm_{3} \).

Large grey arrows, denoted with \( \varepsilon^{3,2} \), represent instantiations of modeling paradigms into metamodels.

We denote by \( L^0 \), \( L^1 \), \( L^2 \), and \( L^3 \) the instance, model, metamodel, and meta-metamodel layers, respectively. Consistently with this notation, we will attach a superscript (from 0 to 3) to each element that is localized on the corresponding layer of the metamodeling architecture.

An example using ADL (Architecture Description Language) Medvidovic & al. [5] describe the C2-style Architecture Description Language in English: “connectors transmit messages between components, while components maintain state, perform operations, and exchange messages with other components via two interfaces (named "top" and "bottom"). ... Inter-component messages are either requests for a component to perform an operation or notifications that a given component has performed an operation or changed state”. Let us call by \( m_{p2} \) the described modeling paradigm. As depicted in Figure 1, the description of \( m_{p3} \) includes concepts (e.g., connector, component, interface, message) and constraints (e.g., “components may not directly exchange messages; they may only do so via connectors”).

Robbins & al. [11] propose an extension of UML’s metamodel for C2-ADL. Their extension is an instantiation of the Medvidovic’s modeling paradigm in terms of a metamodel which we denote by \( mm_{3} \). They define \( \ll C2 - \text{inter face} \rr \) as a stereotype of the UML interface with a tagged value (top, bottom). \( \ll C2 - \text{operation} \rr \) (which can be either a request for operation or a notification message) is defined as a stereotype of the UML operation with

\[ ^{\text{A formal description of C2 in language Z is given in [6].}} \]
Subsumption of modeling paradigms is intended to express a relationship between a modeling paradigm and one of its variants. For example, Koch & al.’s modeling paradigm for hypermedia design [4] subsumes Fröhlich & al.’s modeling paradigm [3] which imposes more explicit restrictions, namely a restriction allowing only binary associations, and an inter-diagram constraint (a limitation of types of navigation—in the navigational structure diagram—depending on multiplicities of associations in the navigational space diagram).

A modeling paradigm \( mp_1 \) is subsumed by a modeling paradigm \( mp_2 \) if both extended inclusion of concepts and subsumption of constraints are satisfied. Extended inclusion of concepts means that each concept of \( E^3(mp_2) \) is either a member of \( E^3(mp_1) \) or a generalization of a concept of \( E^3(mp_1) \), where a generalized concept may have fewer features than its specialization has. Subsumption of constraints means that by using \( C^3(mp_1) \) as a hypothesis, it is possible to prove that each constraint of \( C^3(mp_2) \) holds.

Given two modeling paradigms \( mp_1 \) and \( mp_2 \), we denote by \( mp_1 \preceq mp_2 \) the subsumption of \( mp_1 \) by \( mp_2 \). We denote by \( E^3(mp_2) \subseteq E^3(mp_1) \) the extended inclusion of concepts of \( mp_2 \) in concepts of \( mp_1 \). We denote by \( C^3(mp_1) \subseteq C^3(mp_2) \) the subsumption of constraints of \( mp_1 \) by constraints of \( mp_2 \).

For example, there is an extended inclusion (which we denote by \( \{ t2 \} \subseteq \{ t1 \} \) between a concept of time \( t2 \) with features \( model, unit \) and a concept of time \( t1 \) with features \( model, unit, interpolation \) [10].

We note that the subsumption relation \( \preceq \) is a partial order relation. Given two modeling paradigms \( mp \) and \( mp' \) such that \( mp \preceq mp' \), we use our definition of subsumption of modeling paradigms \( (mp \preceq mp') \) for partitionning the sets of concepts and constraints of the subsumed modeling paradigm \( (mp) \). As an example, we will consider the above modeling paradigm for C2-ADL, \( mp_3 \), which is subsumed by the general modeling paradigm \( gmp \). We obtain:

- A partition of the set of elementary concepts of \( mp \) into three subsets:
  - The set of common concepts: \( \text{Com}(mp, mp') \) contains all concepts belonging to the intersection \( E^3(mp) \cap E^3(mp') \). For example, the concept of class is common to \( mp_3 \) and \( gmp \).
  - The set of specialized concepts \( \text{Spe}(mp, mp') \) contains all concepts of \( mp \) that specialize concepts of \( mp' \). For example, the \( gmp \)’s concept of interface is specialized in \( mp_3 \): “A C2 interface has a tagged value identifying its position. All C2 interface operations must be C2 operations”.
  - The set of new concepts that contains all other concepts of \( mp \): \( \text{New}(mp, mp') = E^3(mp) \setminus (\text{Com}(mp, mp') \cup \text{Spe}(mp, mp')) \) For example, the concept of message is a new concept which is introduced in \( mp_3 \).

- A partition of the set of constraints of \( mp \) in three subsets:
  - The set of shared constraints: \( \text{Sh}(mp, mp') \) that contains all constraints belonging to \( C^3(mp) \cap C^3(mp') \). For example, the constraint “Any object belongs to a class” is common to \( mp_3 \) and \( gmp \).
  - The set \( \text{Ded}(mp, mp') \) of constraints of \( mp \) that are used to deduce non-shared constraints of \( mp' \).

For example, the constraint “Both ends of a C2 attachment must be to a C2 component” of
\( C^3(mp) \) implies that “Both ends of a C2 attachment are to a class” since C2 attachments are stereotypes of class.

- The set of new constraints that contains all other constraints of \( mp \): 
\[
\text{Add}(mp, mp') = C^3(mp) - (Sh(mp, mp') \cup \text{Ded}(mp, mp'))
\]

For example, the constraint “Each C2 component and connector has exactly one instance” is a new constraint which is introduced into \( mp_3 \) and \( gmp \).

**Definition of an instantiation function**  
An instantiation function

\[
\mathcal{E}^{3.2}: \text{Restrict}_{MP} \rightarrow \{\text{UML - metamodel extensions}\}
\]

is defined for building metamodels from modeling paradigms. Let us consider a modeling paradigm \( mp \). \( \mathcal{E}^{3.2} \) associates each concept of \( \mathcal{E}^3(mp) \) with one or more elementary components of the UML language. Such components are either standard UML constructs or stereotypes. We further assume that \( mp \)’s corresponding metamodel \( mm = \mathcal{E}^{3.2}(mp) \) is described by a set of UML-constructs and a set of OCL-constraints (denoted by \( \mathcal{E}^2(mm) \) and \( C^2(mm) \), respectively). Some constraints of \( C^3(mp) \) are included into stereotype definitions. For example, the constraint “messages have no return value” is included in stereotype \( \mathcal{C}2 - \text{operation} \). Thus \( C^2(mm) \) contains instantiations of all other constraints of \( C^3(mp) \), as well as additional constraints due to the instantiation process itself.

**Partitionning of UML-constructs and OCL-constraints of a metamodel**  
Let us consider two metamodels \( mm_1 \) and \( mm_2 \) where \( mm_1 \) is a direct heir of \( mm_2 \). Due to our mirroring structure, their corresponding modeling paradigms \( mp_1 \) and \( mp_2 \) are related by our partial order \( \preceq \) (i.e., \( mp_1 \preceq mp_2 \)). Thus we have –as defined above– a partition of the concepts of \( mp_1 \): \( \text{Com}(mp_1, mp_2) \), \( \text{Spe}(mp_1, mp_2) \), and \( \text{New}(mp_1, mp_2) \); and a partition of the constraints of \( mp_1 \): \( \text{Sh}(mp_1, mp_2) \), \( \text{Ded}(mp_1, mp_2) \), and \( \text{Add}(mp_1, mp_2) \).

By applying the instantiation function \( \mathcal{E}^{3.2} \) to concepts of \( mp_1 \), we induce several subsets of UML-constructs and OCL-constraints. These subsets are depicted in Figure 2. Large grey arrows represent instantiations of concepts and constraints. Double-headed arrows represent inclusion of some constraints of a modeling paradigm in stereotypes of its corresponding metamodel. A thin dark-grey arrow represents instantiation of new concepts/constraints as part of a stereotype defined for specialization: an example is given in Section 3 where two new concepts (for C2-ADL modeling paradigm \( mp_3 \)), namely connector and component, are instantiated as specialization of UML-constructs (as stereotypes of class).

We obtain:

- A subset of constructs of \( mm_1 \) containing all common constructs. This subset is defined by \( \text{Com}_{\text{UML}}(mm_1, mm_2) = \mathcal{E}^{3.2} (\text{Com}(mp_1, mp_2)) \).

- A subset of all pairs of constructs of \( mm_1 \) and \( mm_2 \) (i.e., a subset of \( \mathcal{C}(mm_1) \times \mathcal{C}(mm_2) \)), denoted by \( \text{Spe}_{\text{UML}}(mm_1, mm_2) \), which contains all pairs of specialized-generalized constructs such that the specialized construct of the pair belongs to \( mm_1 \) and the generalized construct of the pair belongs to \( mm_2 \). These pairs of constructs correspond either to the instantiation of specialized concepts (which belong to \( \text{Spe}(mp_1, mp_2) \)) or to some of the additional elementary concepts (which belong to \( \text{New}(mp_1, mp_2) \)).

- A subset of the constructs of \( mm_1 \), denoted by \( \text{New}_{\text{UML}}(mm_1, mm_2) \), containing the new constructs introduced by the instantiation of \( mp_1 \) into \( mm_1 \). These new constructs instantiate some of the additional elementary concepts (which belong to \( \text{New}(mp_1, mp_2) \)).

By applying the same approach to constraints of \( mp_1 \), we induce three subsets of the constraints of \( mp_1 \): the subset of shared constraints \( \text{Sh}_{\text{UML}}(mm_1, mm_2) \); the subset of constraints \( mp_1 \) which are used to deduce constraints of \( mp_2 \), namely \( \text{Ded}_{\text{UML}}(mm_1, mm_2) \); and the subset of new constraints \( \text{Add}_{\text{UML}}(mm_1, mm_2) \). This construction is more complex than the construction of subsets of concepts since a constraint of \( mm_2 \) can be deduced from several constraints of \( mm_1 \). We do not give full details in this paper since the only subset we will use is \( \text{Add}_{\text{UML}}(mm_1, mm_2) \).
3 Semantical distance between metamodels

In order to be able to evaluate semantical quality, we need to measure semantical distances between metamodels. Our proposal for such an evaluation is to build a measure of semantic distance as a weighted sum of elementary distances between corresponding elements of metamodels (i.e., corresponding constructs and corresponding constraints). The main difficulty in such an approach is how to determine corresponding pairs of elements. For that, we use subsets of UML-constructs and of OCL-constraints induced by our subsumption relationship and instantiation function.

In the rest of this section, we provide a generic method for defining a measure of semantical distance between two metamodels. Elementary distances between paired elements, as well as weights used for combination of these elementary distances, are not discussed in the general case: they need to be fine-tuned in the context of a specific application domain.

Let us consider two metamodels \( \text{mm}_1 \) and \( \text{mm}_2 \) and their corresponding modeling paradigms \( \text{mp}_1 \) and \( \text{mp}_2 \), respectively. We assume that \( \text{mm}_1 \) is a direct heir of \( \text{mm}_2 \) (i.e., \( \text{mp}_1 \preceq \text{mp}_2 \)). Our strategy for measurement of semantical distance is defined separately for each of the subsets of UML-constructs:

- Elementary measure due to \( \text{Com}_{\text{UML}}(\text{mm}_1, \text{mm}_2) \) is based on the fact that common constructs implement common concepts. The only variation may be in the way common concepts have been translated by \( \varepsilon^{3.2} \) into UML-constructs. This part of the measure is a syntactical measure and can be ignored in most cases.

- Elementary measure due to \( \text{Spe}_{\text{UML}}(\text{mm}_1, \text{mm}_2) \): we work with pairs of constructs corresponding to a general concept and a specialization of this general concept. For every such a pair, evaluation of distance has to take into account both semantical variations due to added features and syntactical variations (due to \( \varepsilon^{3.2} \)-instantiation).

- Elementary measure due to \( \text{New}_{\text{UML}}(\text{mm}_1, \text{mm}_2) \): new components correspond to concepts that have been introduced in \( \text{mp}_1 \), thus we have to evaluate the semantical cost of the concept itself. We believe that –in most cases– such evaluation can rely on the semantical distance between the new construct and the initial UML-construct that it originates from.

Let us present an example using metamodels \( \text{mm}_3 \) and \( \text{mm}_{\text{UML}} \) (see Figure 1). Metamodel \( \text{mm}_3 \) is a specialization of \( \text{mm}_{\text{UML}} \), and \( \text{mp}_3 \) is subsumed by \( \text{gmp} \). In order to evaluate a semantical distance between \( \text{mm}_3 \) and \( \text{mm}_{\text{UML}} \), we build the following sets:

- At the meta-metamodel level, the set of common concepts:
  \[
  \text{Com}(\text{mp}_3, \text{gmp}) = E^{3.2}(\text{gmp})
  \]

- At the meta-metamodel level, the set of specialized concepts:
  \[
  \text{Spe}(\text{mp}_3, \text{gmp}) = \{ \text{interface}, \text{message} \}
  \]

- At the meta-metamodel level, the set of new concepts:
  \[
  \text{New}(\text{mp}_3, \text{gmp}) = \{ \text{connector}, \text{component} \}
  \]

- At the metamodel level, the pairs of initial UML constructs and specialized constructs:
  \[
  \text{Spe}_{\text{UML}}(\text{mm}_3, \text{mm}_{\text{UML}}) = \{(\text{\langle C2 - operation \rangle}, \text{operation}),
  (\text{\langle C2 - interface \rangle}, \text{interface}),
  (\text{\langle C2 - component \rangle}, \text{class}),
  (\text{\langle C2 - connector \rangle}, \text{class})\}
  \]

We note that the two new concepts of \( \text{mp}_3 \) are instantiated as specializations of the UML class: their instantiations belong to \( \text{Spe}_{\text{UML}}(\text{mm}_3, \text{mm}_{\text{UML}}) \). Thus, \( \text{New}_{\text{UML}}(\text{mm}_3, \text{mm}_{\text{UML}}) = \emptyset \).

Analogously to the above, we use subsets of UML constraints of \( \text{mm}_1 \) and \( \text{mm}_2 \). We assume that the distance due to shared constraints, as well as to deduced constraints, is not significant\(^3\). We consider several types of constraints: 1) constraints which introduce a slight variation (e.g., \( \text{\langle C2 - operation \rangle} \) are limited to have only \( \text{\langle C2 - operation \rangle} \)); 2) weak constraints which cannot weaken a fundamental feature (e.g., a constraint “\( \text{\langle C2 - operation \rangle} \) has no return value” added to the UML-construct \( \text{operation} \) to define a \( \text{\langle C2 - operation \rangle} \) stereotype); and 3) strong constraints which modify a fundamental feature of the corresponding element (e.g., a constraint “each \( \text{\langle C2 - component \rangle} \) has exactly one instance in the running system” induces a major change when going from the UML-construct \( \text{class} \) to the stereotype \( \text{\langle C2 - component \rangle} \)).

We have shown how to determine pairs of corresponding elements (either concepts or constraints) for measurement of a semantical distance between two metamodels when one of the metamodels is a heir of the other. In the general case, we define a more complex partition of sets of constructs and constraints (hints about such a partition can be found in [12]).

\(^3\)This assumption relies on the fact that modeling paradigms are defined by modelers who use OCL as a common foundation for expression of constraints.
4 Conclusion

Our metamodeling architecture takes advantage of the knowledge of modelers’ behaviors, abstract approaches to information system engineering, and formal methods. Modeler’s behavior is represented by a poset of modeling paradigms at the meta-metamodel level. Abstraction in information system engineering is represented by metamodels which also form the core of our architecture and support formal methods. Cohesion of the two uppermost layers of our architecture (i.e., modeling paradigms and metamodels) is guaranteed by the hypothesis of one-to-one correspondence of modeling paradigms and metamodels as well as by mirroring structure of the poset of modeling paradigms and the inheritance hierarchy of metamodels. Such a hypothesis is somewhat restrictive but it allows to “induce” properties of modeling paradigms from properties that have been formally defined on metamodels.

We present formal operations on metamodels. We then extend these operations into semantical ones: semantical integration of metamodels and a measure of a semantical distance between metamodels. These semantical operations form the basis of two frameworks which employ metamodeling in the context of interoperability of information systems and of the Semantic Web. We continue our work on several technical issues related to this project:

- Providing modelers with a convenient interface for using our mirroring structure in the form of a library of domain descriptions (for describing modeling paradigms and instantiating corresponding metamodels). One of the more important part of such an interface is a support for analysis of modeling paradigm descriptions. Such an analyzer must accept as input multi-notation descriptions (which encompass a natural language). It must be able to extract from these descriptions concepts (and constraints) that form the modeling paradigm.

- Providing domain experts with a methodology for adaptation of our generic measure of a semantical distance between metamodels to domain experts’ application domains.

- Defining generic rules for quality measurement of the sub-poset of modeling paradigms that correspond to closely related application domains.

Finally, we believe that the extensive use of our proposal would open a “standardization” issue, namely the need for a new organization of domain modeling. Modelers would be responsible for “local semantics” (i.e., for describing their own application domain as a variation of an existing domain description). Domain experts would be responsible for the global semantics (i.e., for validating semantical dependencies between domain descriptions).

References


