Metamodels and Information System Engineering: a UML-based Approach

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. Metamodels play a major role in most modeling environments. Motivated by a survey of modelers' practice, we show that metamodels are not a suitable description media for modelers. We propose to build a UML-based metamodeling architecture which encompasses modeling paradigms (multilingual and informal descriptions) and metamodels (monolingual and formal descriptions).

1 Introduction

Metamodels are intensively used in information system engineering. In particular, information system modeling uses metamodels for implementation of two abstraction mechanisms (abstraction by projection and abstraction by conceptualization) with an emphasis on abstraction by projection. The Model Driven Architecture approach emphasizes abstraction by conceptualization by using metamodels as classical modeling uses models.

Abstraction by projection relies upon the principles of separation and combination of concerns [1]. Separation of concerns is implemented through a multi-view model which encompasses orthogonal views of the system. A "classical" example of separation of concerns is given by UML's diagrams which offer four views of an information system (user, structural, behavioral, and physical views). A representative description of separation of concerns can be found in Koch & al.'s UML extension for hypermedia [7] in which additional diagrams are defined for modeling of navigation within a web site. Combination of concerns must guarantee that views are consistent which each other and that each piece of information appears in at least one of the views.

Abstraction by conceptualization strives to structure a given description in four layers (instance, model, metamodel, and meta-metamodel layers) which we discuss below. These layers form a metamodeling architecture. The instance and model layers have been used in database modeling for a long time. The metamodel layer defines which language (e.g., which modeling constructs) will be used for modeling of a specific application domain. Representative examples of modeling construct definitions can be found in UML extensions for synchronization [6] and Architecture Description Languages [9]. The meta-metamodel layer describes the "universe of the discourse": how the real world is seen (e.g., time models or spatial models), which languages are used for description of the real world (e.g., boolean or modal logics), etc. Let us mention two areas (namely Model Driven Architecture and interoperability of information systems) in which abstraction by conceptualization has been applied. The Model Driven

Architecture approach moves from a *one model-multiple code components* strategy to a *one metamodel-multiple model components* strategy. Such a transition towards multiple models (which lie at different levels of abstraction) facilitates evolution of information systems. Interoperability of information systems has proposed metamodel-level approaches. These approaches for interoperability determine main concepts of each modeling language (e.g., O.O. classes and associations, relational tables and keys, etc) and they organize these concepts into a common structure (e.g., an inheritance hierarchy, a graph, a lattice). See [8] for a survey of the state of the art. Such a common structure of modeling concepts lies at the metamodel level and forms the basis on which model translation is achieved. Falkenberg & al. [3] have proposed to organize metamodels into an inheritance hierarchy. Such a hierarchy is difficult to build unless we accept certain restrictions on metamodels: in the following, we restrict ourself to metamodels within the scope of the UML metamodel expressiveness.

Many metamodeling architectures use the UML metamodel as a core of both abstraction mechanisms. First, the UML metamodel is used for defining a modeling language which can satisfy the specific needs of a given application domain. Second, the UML metamodel is used for defining the diagrams necessary for description of an application belonging to a given domain.

2 The role of metamodels in information system engineering

The role of metamodels has been discussed for a long time. Finally, a consensus appeared (e.g., with UML and OMG) in which: 1) metamodels form the core of metamodeling architectures, 2) metamodels define languages (or dialects) for model descriptions, 3) metamodels describe the semantics of application domains (metamodels serve as domain ontologies). Metamodels have been also used for model validation [5, 2] and model mapping [10]. We wish to proceed further and thus study the difference between metamodel integration and metamodel translation. In the context of interoperability, it is necessary to provide a semantical basis –in terms of a metamodel– for any kind of cooperation of information systems. Let us employ the context of geographic information systems. Figure 1 presents chunks of cross-domain descriptions which are related to time and space: metamodel-components, model-components, and instance-components.

First, let us assume that all the metamodels under consideration refer to the same time description. Thus, a common semantics for time exists and can be described by an integrated metamodel. In this case, a cross-domain description is based upon the integrated metamodel: transformation tools can be limited to model and instance levels. Thus, semantical integrity of answers to user-queries is guaranteed by the integrated metamodel.

Second, let us assume that metamodels under consideration refer to different space descriptions (Earth space and Galactic space, respectively), and that there is no integrated metamodel for space. In this case, cross-domain description needs to provide transformation tools for space description at the metamodel, model and instance levels. In this case, some of answers to user-queries depend on transformation tool accuracy: the global system has no own semantics for space.

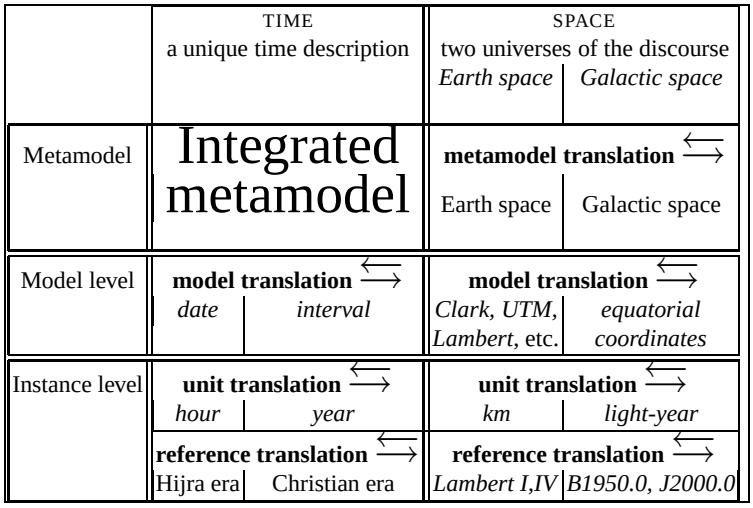

Fig. 1. Metamodels: integration versus translation

Beyond metamodels' role in information system interoperability, metamodel layer appears to be the most promising level for combination of knowledge of modelers' behaviors, abstract approaches to information system engineering, and formal methods. Nevertheless, 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*. As a consequence, metamodels and models do not provide comprehensive information about the actual modeling process: all the initial work (in defining modeling paradigms) has been lost.

We have developped [11, 12] a metamodeling architecture that encompasses modeling paradigms, as well as metamodels and models. Modeling paradigms describe –in terms of concepts that are interrelated by constraints– the semantics that modelers assign to the real world. Modeling paradigms are described informally: their 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, each of them being described with more or less precision. We define a partial order between modeling paradigms by using a subsumption relation: modeling paradigms are organized in a partially ordered set (i.e., a poset). In order to make such a two-fold description (i.e., modeling paradigms and metamodels) meaningful, the two parts of a description must be closely related: the metamodel layer of our architecture is built as a mirror of the poset of modeling paradigms. In this way, metamodels (and modeling paradigms) form the core of application domain descriptions.

3 Conclusion

We believe that the extensive use of metamodels for application domain descriptions [13] would open a "political" 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). In such a case, defining a global structure of metamodels would become soon a major issue in information system engineering.

References

1. Jean Bezivin. On Different Interoperability Modes in Software Engineering: the Case of Modeling Activities at OMG. In *Proc. of Software Engineering'98*, Paris, France, 1998.
2. Aspassia Daskalopulu. Model Checking Contractual Protocols. In *Proc. of the 13th Annual Conference JURIX'2000*, pages 35–47, 2000.
3. E. D. Falkenberg and J.L. Han Oei. Meta Model Hierarchies from an Object-Role Modeling Perspective. In *Proc. of the 1st International Conference on Object-Role Modeling, ORM-1, Magnetic Island, Australia*, 1994.
4. Constance Heitmeyer, James Kirby, Bruce Labaw, and Ramesh Bharadwaj. SCR: A Toolset for Specifying and Analyzing Software Requirements. In *Proc. of the 10th Annual Conference on Computer-Aided Verification, CAV'98*, pages 526–531, Vancouver, Canada, 1998. Available at URL chacs.nrl.navy.mil/SCR.
5. José Luis Herrero, Fernando Sanchez, Fabiola Lucio, and Miguel Toro Bonilla. Changing UML Metamodel in Order to Represent Concern Separation. ECOOP'00 Workshop 14 on Defining a Precise Semantics for UML, Sophia Antipolis, France, 2000.
6. Nora Koch, Hubert Baumeister, Rolf Hennicker, and Luis Mandel. Extending UML for Modeling Navigation and Presentation in Web Applications. In *Proc. of the Workshop Modeling Web Applications in the UML, UML'00*, 2000.
7. Jun Lou. Data Model Description and Translation Using the Meta-model SOME. Phd thesis, EPFL, Lausanne, Switzerland, 1997.
8. Jason E. Robbins, Nemad Medvidovic, David F. Redmiles, and David S. Rosenblum. Integrating Architecture Description Languages with a Standard Design Method. In *Proc. of the 1998 International Conference on Software Engineering*, pages 209–218. IEEE, 1998.
9. J. Sprinkle and G. Karsai. Defining a Basis for Metamodel Driven Model Migration. In *Proc. of the 9th International Conference and Workshop on the Engineering of Computer-Based Systems*. IEEE, 2002. Lund, Sweden.
10. Marie-Noëlle Terrasse. A Metamodeling Approach to Evolution. In H. Balsters, B. de Bruck, and S. Conrad, editors, *Database Schema Evolution and Meta-Modeling*. Springer-Verlag, LNCS 2065, ISBN 3-540-42272-2, 2001. 9th International Workshop on Foundations of Models and Languages for Data and Objects, Schloss Dagstuhl, Germany, September 2000.
11. Marie-Noëlle Terrasse, Marinette Savonnet, and George Becker. An UML-metamodeling Architecture for Interoperability of Information Systems. In *Proc. of the International Conference on Information Systems Modelling, ISM'01*, 2001. Available at URL http://www.fee.vutbr.cz/UIVT/ism/ISM_programElEdition.htm.
12. Dániel Varró and András Pataricza. Metamodeling Mathematics: A Precise and Visual Framework for Describing Semantics Domains of UML Models. In *Proc. of the 5th International Conference on the Unified Modeling Language*. Springer, LNCS 2460, 2002. Dresden, Germany.