What is a Multi-Modelling Language?*

Artur Boronat[1], Alexander Knapp[2], Jose Meseguer[3], and Martin Wirsing[2]

[1] University of Leicester
[2] Ludwig-Maximilians-Universität München
[3] University of Illinois at Urbana-Champaign

1 Abstract

In an ideal software engineering world, development teams would follow well-defined processes in which one single modelling language is used for all requirements and design documents; but in practice "multi-modelling" happens: in a large software project entity relationship diagrams and XML may be used for domain modelling, BPEL for business process orchestration, and UML for design and deployment. UML itself can be seen as a multi-modelling language comprising several sublanguages such as class diagrams, OCL and state machines; each submodelling language provides a particular view on a software system. Such views have the advantage of complexity reduction: a software engineer can concentrate on a particular aspect of the system such as the domain architecture or dynamic interactions between objects.

On the other hand, multi-modelling raises a number of methodological and semantical questions: are the different sublanguages semantically consistent, how can we correctly transform an abstract model of one modelling language into a more concrete one in another language? More generally, is there a notion of "multi-modelling language" which provides more insight than just a bunch of modelling languages together? Is it possible to give a semantics to multi-modelling languages which allows one to deal with consistency, validation and verification but retain the advantages of views by providing a local semantics and local reasoning capabilities for each modelling language?

In the literature, there are three complementary approaches for interrelating modelling notations: the "system model approach", the "model-driven architecture approach", and the "heterogeneous semantics and development approach". In the system model approach the different modelling languages are translated into a common (formally defined) modelling notation called system model [1] which serves as unique semantic basis and for analysing consistency of software engineering models. In the "model-driven architecture approach" [2] model transformations are used for semi-automatically transforming platform-independent models into platform-specific models; consistency questions are typically dealt with at the syntactic level of the modelling notation. In the third approach different modelling languages are interrelated by semantic-preserving mappings [3,4]; a mathematical semantics is given locally for each modelling language and the consistency between different languages is analysed semantically through the semantic-preserving mappings. All three approaches have been applied to several modelling languages including UML, but to our knowledge, multi-modelling languages have never been systematically studied.

* This work has been partially sponsored by the project SENSORIA IST-2005-016004

In this paper we combine ideas from model-driven architecture and heterogeneous semantics and propose a new, semantically well-founded notion of a multi-modelling language and a new notion of semantic correctness for model transformations.

In particular, our formal definition of a multi-modelling language L

- uses the Meta-Object Facility MOF and their algebraic semantics [5] for describing the metamodels and models of the sublanguages of L
- associates an institution to each sublanguage S of L and a gives a mathematical semantics to each software engineering model [4] of S by a corresponding (logical) theory in the institution of S
- defines the links between different sublanguages of S by model transformations and provides a notion of semantic correctness for such transformations
- provides a notion of consistent heterogeneous (software engineering) model of the multi-modelling language L which is derived from a notion of a category of heterogeneous mathematical models at the institution level.

In the full paper we will illustrate these ideas in the context of existing modelling languages by presenting a case study which involves models in several modelling languages, and explain how our concepts can be applied to show the consistency of software engineering models and the semantic correctness of model transformations. In particular, we choose UML and entity relationship diagrams as modelling languages and combine them via a semantically correct model transformations to a multi-modelling language. Based on earlier work [4] we show that class diagrams and OCL form a multi-modelling language where class diagrams are related to OCL by a semantically correct model transformation. Then we obtain the full multimodelling language by a semantically correct model transformation from class diagrams to entity relationship diagrams.

References

1. Broy, M., Cengarle, M.V., Rumpe, B.: Semantics of UML – Towards a System Model for UML: The Structural Data Model. Technical Report TUM-I0612, Institut für Informatik, Technische Universität München (2006)
2. Object Management Group (OMG): MDA Guide Version 1.0.1. Technical report, OMG (2003) `www.omg.org/docs/omg/03-06-01.pdf`.
3. Mossakowski, T.: Heterogeneous Specification and the Heterogeneous Tool Set. Habilitation thesis, Universität Bremen (2005)
4. Cengarle, M.V., Knapp, A., Tarlecki, A., Wirsing, M.: A Heterogeneous Approach to UML Semantics. In: Festschrift for Ugo Montanari. Volume 5019 of Lect. Notes in Comp. Sci., Springer (2008) To appear.
5. Boronat, A., Meseguer, J.: An Algebraic Semantics for MOF. In: FASE 2008, Budapest, Hungary, March 29-April 6, Proceedings. Lect. Notes in Comp. Sci., Springer (2008)

[4] For distinguishing semantic models from the models of a modelling language we write "software engineering model" for a (syntactic) description of a model in a modelling language such as UML. In contrast to this, "(semantic) models" are part of the mathematical semantics of a modelling language and therefore a semantic model can be understood as a model of a theory in a suitable logic.