Chapter 4

Formalization of MOF–Based Metamodels

INTRODUCTION

Formal and semiformal techniques can play complementary roles in MDA-based software development processes. We consider it beneficial for both semiformal and formal specification techniques. On the one hand, semiformal techniques lack a precise semantics; however, they have the ability to visualize language constructions, allowing a great difference in the productivity of the specification process, especially when the graphical view is supported by means of good tools. On the other hand, formal specification allows us to produce a precise and analyzable software specification and clarifies the intended meaning of metamodels, helps to validate model transformations, and provides reference for implementations; however, they require familiarity with formal notations that most designers and implementers do not currently have and the learning curve for the application of these techniques requires considerable time.

A combination of metamodeling and formal specification techniques can help us to address MDA-based processes such as reverse engineering, forward engineering and round-trip engineering. In light of this, we propose to use the algebraic metamodeling language, called NEREUS which can be viewed as an intermediate notation. NEREUS can be integrated with different formal languages and object-oriented languages. It is particularly suited for specifying metamodels based on the concepts of entity, relation and system. Most of the MOF metamodel concepts can be mapped directly to NEREUS.

In terms of NEREUS we will explain a reusable infrastructure for more efficient development of evolution system techniques and high quality of the results. In the following sections we summarize the main specification languages linked to object-orientation approaches such as UML and MDA and, motivate the use of NEREUS as a metamodeling language.

DOI: 10.4018/978-1-61520-649-0.ch004
OBJECT-ORIENTATION, METAMODELING AND FORMAL LANGUAGES

In the early 1980s, new specification languages or extensions of formal languages to support object-oriented concepts began to develop. Among them, the different extensions of the Z language, for example Z++ (Lano, 1991), OBJECT-Z (Smith, 2000) (Kim, & Carrington, 1999) or OOZE (Alencar & Goguen, 1991) can be mentioned. Another language with object-oriented characteristics is FOOPS (Rappanotti & Socorro, 1992).

Larch/Smalltalk was the first language with subtype and inheritance specification. Larch/C++ is another language with similar characteristics. JML is a behavioral interface specification language for formally specifying the behavior and interfaces of Java classes and functions (Leavens, 1996) (Leavens et al., 2002).

CASL-LTL, an extension of CASL (Bidoit & Mosses, 2004), has been provided to deal with reactivity (Reggio, Cerioli & Astesiano, 2001).

BON is an object-oriented method possessing graphical and textual languages for specifying classes, their relations and assertions, written in first-order predicate logic (Paige, Kaminskaya & Ostroff, 2002).

Various works analyzed the integration of semiformal techniques and object-oriented designs with formal techniques. (Bordeau, 1995) introduces a method to derive Larch specifications from class diagrams. France, Bruel and Larrondo-Prieti (1997) describe the formalization of FUSION models in Z.

A lot of work has been carried out dealing with the semantics for UML/OCL models. The PreciseUML Group, pUML, was created in 1997 with the goal of giving precision to UML (France, Evans, Lano & Rumpe, 1998).

Considering that OCL has only a denotational semantics that can be implemented by dynamic validation of snapshots, several works propose UML formalization by using traditional formal languages. These formal languages provide at least syntax, some semantics and an inference system. The syntax defines the structure of the text of a formal specification including properties that are expressed as axioms, formulas of some logic. The semantics describes the models linked to a given specification; in the formal specification context, a model is a mathematical object that describes behavior of the realizations of the specifications. The inference system allows defining deductions that can be made from a formal specification. These deductions allow new formulas to be derived and checked. So, the inference system can help to automate testing, prototyping or verification.


Borger, Cavarra and Riccobene (2000) provide a rigorous semantics for one of the central diagram types which are used in UML for the description of dynamical system behavior, namely activity diagrams.

Reggio, Cerioli and Astesiano (2001) present a general framework of the semantics of UML, where an individual semantics is given to the different kinds of UML diagrams within a UML model, and next, such semantics are composed to get the semantics on the overall model.

Gerber et al. (2002) explores the state-of-the-art of model-to-model transformation. Strengths and weaknesses of different technologies such as Common Warehouse Metamodel (CWM) transformations (CWM, 2003) and graph transformations (Mens et al., 2006) are remarked.