Chapter 9
A Rigorous Approach for Metamodel Evolution

Claudia Pereira
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

Liliana Favre
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina & Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Argentina

Liliana Martinez
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

ABSTRACT

Model-Driven Development (MDD) is an initiative proposed by the Object Management Group (OMG) to model centric software development. It is based on the concepts of models, metamodels, and automatic transformations. Models and metamodels are in constant evolution. Metamodel evolution may cause conforming models to become invalid. Therefore, models must be migrated to maintain conformance to their metamodels. Metamodel evolution and model migration are typically performed manually, which is an error-prone task. In light of this, the authors propose a framework for metamodel evolution and model migration that combine an operator-based approach with refactoring and specification matching techniques. They combine classical metamodeling techniques with formal specifications to reason about transformations, metamodels, and their evolution. The authors describe foundations for MDA-based metamodel evolution that allow extending the functionality of the existing CASE tools in order to improve the MDA-based process quality.

INTRODUCTION

Model-Driven Development (MDD) is a software development approach in which models are considered first-class entities. The goal is to raise the level of abstraction and use concepts closer to the problem and application domain rather than the solution and technical domain. As a result, there is an increasing need for effective techniques and tools to maintain and evolve the models.

Model-Driven Architecture (MDA) (MDA, 2012) is the most known realization of MDD

DOI: 10.4018/978-1-4666-4494-6.ch009

proposed by the Object Management Group (OMG). The key idea behind MDA is to separate the specification of the system functionality from its implementation on specific platforms, managing the software evolution from abstract models to implementations increasing the degree of automation and achieving interoperability with multiple platforms, formal languages and programming languages. Artifacts generated during software development are represented using common metamodeling languages. Metamodeling is an important technique for defining modeling languages. Modeling languages allow transformations can be performed automatically by tools that can understand models. Besides, the transformation rules use the metamodels of the source and target language to define transformations relating elements of the respective languages.

OMG has established different technologies that enable model-driven approach. The object-oriented models used in MDA can be expressed using UML (UML, 2011a; UML, 2011b). Models can be enriched with OCL expressions which specify invariant conditions that must hold for the system being modeled or queries over objects described in a model (OCL, 2012). As a result, models are more accurate and complete. Metamodels in MDA are specified with MOF (MetaObject Facility) (MOF, 2011).

Metamodels, like any software artifact, are subject to constant evolution during its life cycle. This evolution may be due to changes produced during the metamodel definition to improve or correct the abstract syntax of the language, as well as changes in requirements and technological progress in specific application domain. These modifications include changes at all levels, from requirements through architecture and design, to executable models, documentation and test suites. Changes typically affect various kinds of models including data models, behavioral models, domain models, source code models and goal models.

Changes introduced in a metamodel can be classified by their corrupting or non-corrupting effects on existing models (Becker, Gruschko, Goldschmidt, & Koziolek, 2007):

- **Non-Breaking Changes:** Changes which do not break the conformance of models to the corresponding metamodel;
- **Breaking and Resolvable Changes:** Changes which do break the conformance of models, even though they can be automatically resolved;
- **Breaking and Not Resolvable Changes:** Changes which do break the conformance of models which cannot automatically resolved and user intervention is required.

Metamodel evolution can invalidate the models that conform to its previous version. To avoid building models from scratch, they should be migrated in order to conform to the modified metamodel. The activity of changing the metamodel together with its models is called co-evolution.

Model migration, like metamodel evolution, is typically performed manually. This is an error-prone task leading to inconsistencies between the metamodel and related artifacts. Therefore, it is necessary to automate this process using tools that support MDD. This task requires the definition of verification and validation techniques for the effective development of metamodels and the definition of rigorous techniques for migrating models.

Commercial MDD CASE tools provide support for the construction of metamodels and its models as well as mechanisms to check the conformance between models and its corresponding metamodel. However, tools do not support the automation of metamodel evolution. As a result, models become invalid if the metamodel evolution break model conformance. On the other hand, CASE tools do not integrate tools such as automated theorem