

Software Transformation Engineering

Benoit Baudry and Sébastien Gérard

CEA-List, 91191 Gif-sur-Yvette cedex, France
{benoit.baudry, sebastien.gerard}@cea.fr

MDA proposes a move away from human interpretation of high-level models, such as design diagrams, into implementations, towards a more automated process where the models are used as first-class artifacts of the development process. Programs manipulating models (model transformations) then become the fundamental tools for software development. The development of transformations thus has to be based on sound software engineering methods and tools.

We realize now that a usual development process can be applied to transformation development. However, model transformations have particular requirements and structures that make necessary the adaptation of usual software engineering steps and techniques. We believe these adaptations are crucial issues that must be solved in a rigorous way to develop large and robust transformations. In the following, we go through the three steps of specification, design and validation and identify specificities for model transformation development.

The specification of a model transformation should consist at least of pre and post conditions. The pre condition should serve two purposes: define expected properties on input models that specify the licit models for the transformation; express information on the actual subset of the metamodel that is concerned by the transformation. The post condition expresses both the expected result of the transformation and the relationship between source and target models. Concerning the latter point, the OMG RFP for QVT states that the transformation definition language should be able to describe the relationship between source and target metamodel elements.

Several issues appear to design model transformation. First, it is not possible to model a model transformation, because there is no agreement on any abstract syntax. Other issues for designing model transformations are: composition and decomposition of model transformations; the distribution of model transformations; the definition of design patterns.

Today, the most common technique for validation remains testing. In the same way it had to be adapted when switching from procedural to OO programming, we believe it has to be adapted to the model-based context. Several points can be investigated to test model transformations: category-partitioned testing can be adapted by defining coverage criteria on the metamodel; structural testing can also be adapted for transformation languages; fault models should be identified for transformations to adapt mutation analysis.

Once a model transformation is specified, designed and validated, it is used by software developers who develop an application in a model-based context. A nice framework for model transformation usage would be a three-view model: an abstract view (abstract syntax), a user-oriented view (a wizard) to fix the parameters for the transformation and a concrete, runnable view (the transformation program).