# Validated Model Transformation

**Abstract**. Model-Driven Architecture (MDA) as a model-based approach to software development facilitates the synthesis of application programs from models created using customized, domain-specific model processors. MDA model compilers can be realized by graph rewriting-based model transformation. In Visual Modeling and Transformation System (VMTS), metamodel-based transformation steps enable assigning OCL constraints to model transformation steps. Based on this facility, we propose a validated model transformation approach that can validate not only the individual transformation steps, but the whole transformations as well.

## *1 Introduction*

A Model-driven development approaches (e.g. Model-Integrated Computing (MIC) [1] and OMG's Model-Driven Architecture (MDA) [2] emphasize the use of models at all stages of system development. They have placed model-based approaches to software development into focus. MIC advocates the use of domain-specific concepts to represent the system design. Domain-specific models are then used to synthesize executable systems, perform analysis or drive simulations. Using domain concepts to represent the system design helps increase productivity, makes systems easier to maintain, and shortens the development cycle.
The approach presented here uses graph rewriting-based visual model transformation. Models can be considered special graphs, which contain nodes and edges between them. This formal background facilitates to treat models as labeled graphs and to apply graph transformation algorithms to models using graph rewriting. To define the transformation steps precisely and support the validated model transformation beyond the structure of the visual models, additional constraints must be specified which ensure the correctness of the attributes, or other properties can be enforced. Using Object Constraint Language (OCL) [3] constraints provides a solution for these issues. The use of OCL as a constraint and query language in modeling is found to be simple and powerful.
The main contribution of the current paper is the validated online model transformation. The rest of this work is organized as follows: The motivation is presented on a real word case study (Section 2). Section 3 introduces the principles of the validated model transformation based on the relation between the pre- and postconditions and OCL constraints propagated to model transformation steps. The approach presented here makes possible to require single transformation steps as well as the whole transformations to validate, preserve or guarantee certain properties during the transformation. Finally, conclusions are provided.

## *2 Motivation – A Case Study*

To illustrate the motivations on a real word example a case study is provided. The case study is a variation of the "class model to relational database management system (RDBMS) model" transformation (also referred to as object-relational mapping). A database design is often referred to as a data model or schema.
The requirements stated against the transformation that it should guarantee the following properties:

- Classes that are marked as non-abstract in the source model should be transformed into a single table of the same name in the target model. The resultant table should contain one added primary key column, one or more columns for each attribute in the class, and one or more columns for associations based on the next rule.

- In general, an association may, or may not, map to a table. It depends on the type and multiplicity of the association.
  - Many-to-many (N:N) associations, should be mapped to distinct tables. The primary keys for both related classes should become attributes of the association table (foreign keys). Foreign keys do not allow NULL values, because a link between two objects requires that both of them should be known.
  - One-to-many (1:N) associations using one or more foreign key columns should be merged into the table for the class on the “many” side.
  - For one-to-one (1:1) associations, the foreign key should be buried optionally in one of the affected tables.
- Parent class attributes should be mapped into tables created from inherited classes.
- An association class should be transformed based on the multiplicity of the association. For N:N associations the attributes of the association class become columns of the distinct table. For a 1:N or 1:1 the attributes of the association class become columns of the table in which the foreign key is buried.

The required rules jointly guarantee that the generated database is in third normal form [4].

At the implementation level, system validation can be achieved by testing. Various tools and methodologies have been developed to assist in testing the implementation of a system (for example, unit testing, mutation testing, and white/black box testing). However, in case of model transformation environments, it is not enough to validate that the transformation engine itself works as it is expected. The transformation specification should also be validated.

In the case of the case study the following issues should be guaranteed by the transformation: (i) Each table has a primary key, (ii) each class attribute is part of a table, (iii) each parent class attribute is part of a table created for its inherited class, (iv) each many-to-many association has a distinct table, (v) each one-to-many and one-to-one association has merged into the appropriate tables, (vi) foreign keys do not allow NULL value, and (vii) each association class attribute is buried into the appropriate table based on the multiplicities of its association.

There is a need for a solution that can validate model transformation specifications: online validated model transformation that guarantees if the transformation finishes successfully, the generated output (database schema) is valid, and it is in accordance with the requirements above.

## 3 Contributions

Graph rewriting [5] is a powerful technique for graph transformation with a strong mathematical background. The atoms of graph transformations are rewriting rules, each rule consists of a left-hand side graph (LHS) and right-hand side graph (RHS). Applying a graph rewriting rule means finding an isomorphic occurrence (match) of LHS in the graph the rule being applied to (host graph), and replacing this subgraph with RHS.

The Object Constraint Language is a formal language for the analysis and design of software systems. It is a subset of the UML standard [6], and OCL allows software developers to write constraints and queries over object models.

A *precondition* assigned to a transformation step is a Boolean expression that must be true at the moment when the transformation step is fired. Similarly, a *postcondition* assigned to a transformation step is a Boolean expression that must be true after the completion of a transformation step. If a precondition of a transformation step is not true then the transformation step fails without being fired. If a postcondition of a transformation step is not true after the execution of the transformation step then the transformation step fails. A direct corollary of this is that an OCL expression in LHS is a precondition to the transformation step, and an OCL expression in RHS is a postcondition to the transformation step. A transformation step can be fired if and only if all conditions enlisted in LHS are true. Also, if a transformation step finished successfully then all conditions enlisted in RHS must be true [7].

There are three properties: *validation, preservation,* and *guarantee* [5], these properties are checked during the rewriting process. A transformation step *S validates* a property *P* if the following condition always holds: if a property *P* was true before the step *S* it remains true after the execution of the step *S*, and if *P* is false, the step *S* fails. A step *S preserves* a property *P*, when the following condition always holds: if a property *P* was false (true) before the step *S* it remains false (true) after the execution of the step *S*. A transformation step *S guarantees* a property *P*, when the following condition always holds: if a property *P* was true before the step *S* it remains true after the execution of the step *S*, and if *P* is false, the step *S* changes property *P* to true.

| | **property *P* before the step *S*** | **property *P* after the step *S*** |
|---|---|---|
| **Validation** | true | true |
| | false | step *S* fails |
| **Preservation** | true | true |
| | false | false |
| **Guarantee** | true | true |
| | false | true |

Table 1: Truth table of the validation, preservation and guarantee properties

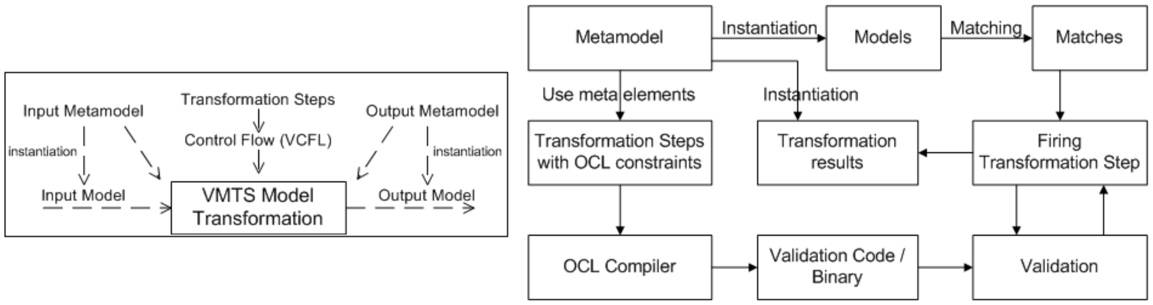

Figure 1: Principles of VMTS metamodel-based validated model transformation

The principles of metamodel-based validated model transformation in VMTS are depicted in Fig. 1. The figure describes that the transformation is specified by the VCFL control flow model that defines the exact execution order of the transformation steps. The input model is described by the input metamodel, and the output model by the output metamodel. Both input and output metamodels have an effect on the transformation.

Recall that LHS and RHS of a transformation step are built from the metamodel elements. It is possible that LHS and RHS use different metamodels, but, for the sake of simplicity, they have a common metamodel in the block diagram (Fig. 1). The transformation step contains OCL constraints. The transformation uses matches found by the matching process and the compiled binary to validate the constraints on the matched parts of the input model. If and only if a match satisfies the constraints (preconditions), then the transformation generates the transformation result, and if and only if the transformation result satisfies the postconditions, then the step was successful. In Fig. 1, the transformation result is also an instance model of the metamodel, because LHS and RHS use the same metamodel.

The constraints assigned to the transformation steps guarantee the requirements expected from the transformation steps, e.g. the requirements from Section 2. As it was discussed, after a successful step execution the conditions hold, and the output is valid. This cannot be achieved without constraints.

## 4 Further Issues

The relationship between the pre- and postconditions and OCL constraints have been shown. The main result of the paper is illustrating online validated model transformation that applying OCL constraints propagated to transformation steps facilitates to require the transformations to validate, preserve or guarantee certain model properties.

A more efficient solution is if the constraints that require the presented properties are propagated automatically to the transformation steps. This can be achieved applying aspect-oriented constraint management [8]. Further open questions are the following: (i) the traceability of the whole transformation, e.g. with trace objects created during the transformation, (ii) bidirectional transformations in order to support round-trip engineering, and (iii) supporting the model evolution with model transformation-based methods.

## References

[1] J. Sztipanovits, and G. Karsai, Model-Integrated Computing, IEEE Computer, Apr. 1997, pp. 110-112.

[2] OMG MDA Guide Version 1.0.1, OMG, doc. number: omg/2003-06-01, 12th June 2003, http://www.omg.org/docs/omg/03-06-01.pdf

[3] OMG Object Constraint Language Spec. (OCL), www.omg.org

[4] Michael R Blaha, and William Premerlani, Object-Oriented Modeling and Design for Database Applications, Prentice Hall, 1998.

[5] G. Rozenberg (ed.), Handbook on Graph Grammars and Computing by Graph Transformation: Foundations, Vol.1 World Scientific, Singapore, 1997.

[6] OMG UML 2.0 Specifications, http://www.omg.org/uml/

[7] L. Lengyel, T. Levendovszky, H. Charaf, Implementing an OCL Compiler for .NET, In Proceedings of the 3rd International Conference on .NET Technologies, Pilsen, Czech Republic, May-June 2005, pp. 121-130.

[8] L. Lengyel, T. Levendovszky, G. Mezei, B. Forstner, H. Charaf, Metamodel-Based Model Transformation with Aspect-Oriented Constraints, International Workshop on Graph and Model Transformation, GraMoT, ENTCS Vol. 152, Tallinn, Estonia, September 28, 2005, pp. 111-123.