Model Transformations powered by Rewriting Logic *

Francisco J. Lucas and Ambrosio Toval

Software Engineering Research Group
Department of Informatics and Systems
University of Murcia (Spain)
fjlucas@um.es,atoval@um.es

Abstract. This paper shows a rigorous approach based on algebraic specifications and rewriting logic which makes up for the lack of current transformation languages and offers a balanced rigour-versus-intuition framework for model transformation, focusing on the MDA-QVT standards. To illustrate this approach, an example and some formal applications of these specifications are sketched.

1 Introduction

In recent years, the profound impact of the Model Driven Architecture (MDA) proposal [1], promoted by the OMG as architecture for software development, has meant that the model transformation becomes a very active research and development direction. Within this scope, OMG also published the QVT standard [2], a language for the specification of model transformations within the MDA scope. Since transformations guide the whole software development cycle, it is crucial to offer a precise and rigorous infrastructure, in order to help to verify and guarantee correctness of them. However, current implementations of transformations languages lack this necessary mathematical underpinning.

The aim of this short paper is: on the one hand, to show how a rigorous approach based on algebraic specification and rewriting logic [3] can offer a suitable framework for model transformations; and on the other hand, to show how proving theoretical properties of transformations is possible if the transformation language or tool has a mathematical underpinning [4]. *Maude* [5] is the formal language used in this work. To illustrate this approach, the formal specifications of two metamodels, and a transformation between them, have been created.

2 Model Transformation based on a Rewriting Logic Approach

The main idea of this work is to specify metamodels through OO Maude modules, and the transformation rules as rewriting rules that rewrite source model

* Partially financed by the Spanish Ministry of Science and Technology, project DEDALO (Development of Quality Systems based on Models and Requirements) TIN2006-15175-C05-03

elements (represented as terms) into other target model elements. In this section, as well as explaining the main principles of this approach, we will illustrate it by means of an example of transformation from UML Class diagram to a RDBMS diagram (extracted from [2]).

2.1 Metamodel Formalization

In QVT, model transformations are defined in terms of metamodels, existing source and target metamodels. Metamodel elements will be specified in *Maude* by means of the object oriented modules of *Maude*. Figure 1 summarizes the elements that make up the approach. This Figure shows the OMG standards' elements, the *Maude* elements and the relationship between them.

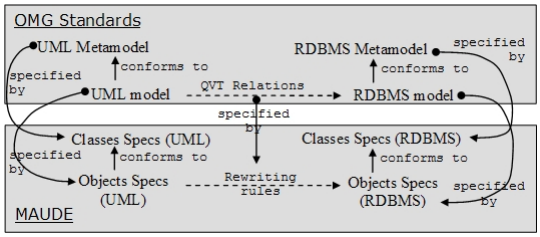

Fig. 1. Summary of the approach elements

The first metamodel specified is the so-called simple UML metamodel (taken from annex A of [2]) which represents a simplified version of the UML Class diagram. Each element of this metamodel is specified by means of a *Maude* class. Figure 2 shows an example of a simple UML diagram and how it is expressed by means of objects in *Maude*; these objects are instances of *Maude* classes that appear in the lefthand side of the figure, as it was indicated in Figure 1.

Analogously, the textual description of the simple RDBMS metamodel (taken from annex A of [2]) will be formalized in *Maude*.

2.2 QVT Relations Features in *Maude*

In this subsection, we analyze briefly the basis of QVT Relations and how to formalize them by means of the strengths offered by *Maude*.

On the one hand, a transformation is expressed in QVT Relations by means of relations between metamodel elements. A relation declares constraints that must be satisfied by the two or more metamodels (or domains) that participate in the relation. Each domain establishes a pattern that must be matched with the candidate models in order to execute the transformation, known as *object template expressions* that are directly expressed in *Maude*, since this language offers pattern-matching in the simplification of terms. Regarding the specification of QVT transformations in *Maude*, they can be specified as rewriting rules that

change and create the elements of the target model. Finally, constraints over the candidate models will be specified as conditions in the rewriting rules.

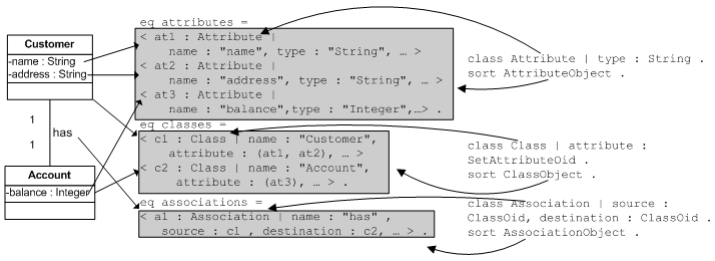

Fig. 2. Example of UML diagram and their corresponding Maude objects

2.3 Simple UML to Simple RDBMS

In this section, we will study the transformation from UML diagrams to RDBMS diagrams (taken from [2]). Basically this transformation has three main (top-level) relations: Package to Schema, Class to Table and Association to Foreign Keys. Due to lack of space only the first relation will be studied.

Package to Schema relation transforms a package of a UML diagram into a schema of a relational data base diagram. Figure 3 shows this relation expressed in QVT (a) and how it is specified in *Maude* (b). In this relation, the object template expression can be "directly" expressed in *Maude*. The pattern matching binds the variable "pn" to a specific value that is used to create a new object which represents a schema in the target model. On the other hand, since a model is represented by means of objects in *Maude*, we have to use object identifiers in the rules. In the righthand side of the rule appears both the object of the source model and the new object in the target model. Finally, once we have specified the metamodels and the QVT Relations in *Maude*, we can execute the transformation over any UML model.

3 Applications in Practice

The main advantage of the use of this approach over other non-formal transformation techniques is that some applications can be carried out only defining the rewriting rules. The application shown checks if two models are semantically equivalent. In general, various models are semantically equivalent if they have similar meaning. Being more precise, in this work we consider that two models are semantically equivalent if they hold all the equivalence relationships (defined by means of QVT Relations) which, depending on the metamodels, express the semantics equivalence of concepts as defined by the analyst. In this way, if the

```
(a) top relation PackageToSchema {  pn: String;
      checkonly domain uml p:Package {name=pn};
      enforce domain rdbms s:Schema {name=pn}; }
(b)
rl [PackageToSchema] :
< packageOid(p:String) : Package | domain : "uml", name : pn:String,...> =>
  < packageOid(p:String) : Package | domain : "uml", name : pn:String,...>
  < schemaOid(p:String) : Schema | domain : "rdbms", name : pn:String > .
(c) (search UMLdiagram =>! C:Configuration C2:Configuration
       such that (C:Configuration := RDBMSdiagram) .)
```

Fig. 3. (a) and (b) Package to Schema (*(a)* adapted from [2]); (c) Maude comand to check if two models are equivalent

semantic equivalence between two models is expressed as a relation, we can use *Maude* to infer if two particular models are equivalent using these rules.

If we define a RDBMS model that is equivalent to the one in Figure 2, Figure 3 (c) shows the *Maude* command that checks this equivalence. We ask *Maude* if it is possible to obtain the *"RDBMSdiagram"* model from the *"UMLdiagram"* model using the specified rules. This execution will find a solution since the models are equivalent.

4 Conclusions

The research presented shows the feasibility of integrating formal techniques with current software engineering standards (MDA-QVT). This approach may be particularly useful in model-driven engineering processes to develop critical or error-prone high quality systems. The metamodel specifications made in this approach offer a powerful way to verify type properties and the correctness of the models without losing the legibility and practicality of other transformation languages. Furthermore, in the formal framework proposed the transformations are represented as mathematical entities and we can take advantage of all the power of mathematical inference mechanisms. This allows us to infer information and to prove properties of the transformations.

References

1. OMG: MDA Guide Version 1.0.1, http://www.omg.org/mda. (2001)
2. OMG: MOF QVT Final Adopted Specification. Object Management Group., Retrieved from: http://www.omg.org/docs/ptc/07-07-07.pdf. (2007)
3. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science, 96(1):73-155 (1992)
4. Mens, T., Czarnecki, K., Van Gorp, P.: A Taxonomy of Model Transformations. Int. Workshop on Graph and Model Transformation (GraMoT). Estonia (2005)
5. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcote, C.: Maude 2.3 Manual., http://maude.csl.sri.com/. (2007)