Towards a Unified Notation to Represent Model Transformation

Anne Etien, Cedric Dumoulin, Emmanuel Renaux

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Towards a Unified Notation to Represent Model Transformation

Anne Etien — Cedric Dumoulin — Emmanuel Renaux —
Email: {etien, dumoulin, renaux}@lifl.fr

N° 6187

May 9, 2007

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6187--FR+ENG

Towards a Unified Notation to Represent Model Transformation

Anne Etien , Cedric Dumoulin , Emmanuel Renaux ,
Email: {etien, dumoulin, renaux}@lifl.fr

Thème COM — Systèmes communicants
Projet DaRT

Rapport de recherche n° 6187 — May 9, 2007 — 14 pages

Abstract: In order to unify our internal exchange and communication about transformations, we propose TrML (Transformation Modeling Language), a unified UML notation to design model transformations. This proposal aims to reify the synthesis of existing notations dedicated to transformation modeling. TrML is independent from implementation details and could be adapted to several transformation engines. To let TrML run on top of existing engine, we transform TrML model to a model accepted by the engine. But, which language should we use for the first transformation? TrML, the targeted engine or another one? In this article we will describe how we bootstrap our new language on top of existing transformation engines.

Key-words: Model transformation, Bootstrap, TrML

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 —Télécopie : +33 1 60 19 66 08

Vers une notation unifiée pour modéliser les transformations de modèles

Résumé : Afin d'uniformiser au sein de l'équipe les échanges et la communication sur les transformations de modèles, nous proposons TrML (Transformation Modeling Language), une notation unifiée pour représenter graphiquement les transformations de modèles. Cette proposition a pour but de concrétiser la synthèse de notations existantes dédiées aux transformations de modèles. TrML est indépendant des détails d'implémentation et peut être adapté pour différents moteurs de transformation. Afin d'utiliser TrML avec des moteurs existant, nous transformons le modèle TrML vers un modèle accepté par le moteur choisi. Maix quel langage doit-on choisir pour écrire la première transformation ? TrML, le modèle cible adapté au moteur ou un autre langage ? Dans cet article, nous décrivons comment 'bootstraper' notre langage au dessus de moteurs de transformation existants.

Mots-clés : Transformation de modèles, Bootstrap, TrML

Introduction

Transformation chains involve several models from high abstraction level to implementation levels. Each abstraction level refers to a specific system viewpoint that an expert is responsible for. Thus to build the whole product chain, experts have to exchange and communicate about transformations. They need an abstract language focusing on the intention of the transformation and that homogenizes documentation.

Since years, modeling activities have been mainly supported by UML-like modeler tools and then, visual notation has become more familiar. Graphical representation of transformation model thus seems to be the best abstraction to focus on the purpose of the transformation independently from the implementation details. Some graphical transformation languages propose a notation visually close to classical representation of the model the transformation applies to. Most of these contributions emerge from graph community [5, 2] and not from the Model Driven Engineering domain that however recommends separation of graphical notation from the implementation language.

We planned empirical studies about the need of a visual notation to represent model transformation independently from implementation platform. After having assessed existing model transformation solutions, we have developed the Transformation Modeling Language TrML. In this modeling language, we gather the core features that we consider mandatory to exchange and communicate about transformation. We provide the corresponding UML profile that makes it portable on any UML tool. TrML is not associated to a transformation engine but is defined in a metamodel allowing to map it on any languages.

Transformations aim to be automatically executed. It is thus essential to provide a transformation engine for TrML transformations. We could have constructed our own transformation engine, but it is contradictory to the essence of TrML that aims to provide a graphical notation independently from the transformation engine or language. Furthermore, there are several transformation engines that have proved their efficiency. We thus propose to build TrML on top of an existing transformation engine. For this purpose, we adopt the bootstrap mechanism allowing to write a transformation from TrML to the model associated to the targeted engine. ATL has been chosen as the targeted engine. However, the principles described in this paper are general and can be applied to other languages such as for example QVT.

The paper is organized as follow. Section 2 describes the basic features of TrML illustrated with the UML to RDBMS example. Section 3 presents the implementation of TrML on top of an existing transformation language and illustrates it with ATL. Section 4 compares TrML to other works on model transformation. Section 5 concludes the article.

1 TrML to graphically design model transformations

This section details TrML core features and illustrates the syntax and the basic semantics of the design representation, specifying UML2RDBMS transformation example. This example is extracted from the QVT specification and describes a simple transformation from a class model to a database schema, summed up in the following sentences:

- *s1. a persistent class maps to a table, a primary key and an identifying column.*
- *s2. attributes of the persistent class map to columns of the table:*
  - *an attribute of a primitive data type maps to a single column;*
  - *an attribute of a complex data type maps to a set of columns corresponding to its exploded set of primitive data type attributes;*
  - *attributes inherited from the class hierarchy are also mapped to the columns of the table.*

- *s3. an association between two persistent classes maps to a foreign key relationship between the corresponding tables.*

1.1 Description of TrML features through an example

Rule. Transformations are mostly too complex to be performed on one shot. They are decomposed into a set of rules that are more legible and allow to reduce the problem to solve. A rule focuses on a small part of the models to be transformed. It is made of input and output patterns describing what should be transformed into what, and how elements are linked (via bind names).

There are different kind of rules: toprules (Figure 1), that are directly executed by the engine; normal rule (Figure 3), explicitly called from another rule; setRules (Figure 2) and listRules, are collections of rules where each rule is executed in any order or in the specified order respectively; selectRule, an ordered set of rules where only the first applicable rule is executed.

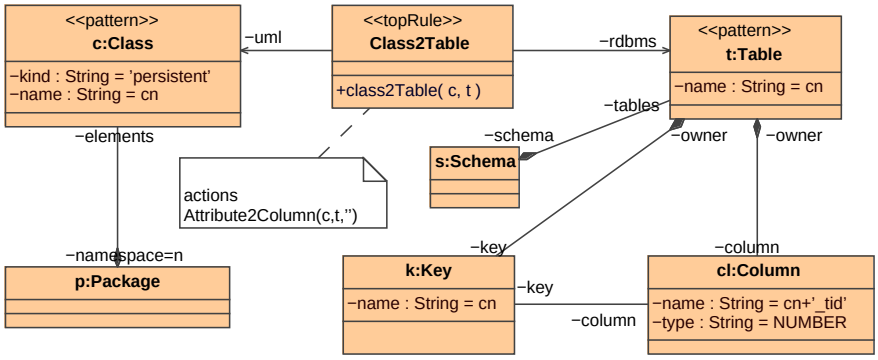

Figure 1: Description of the Class2Table rule

One goal of TrML is to be able to describe a rule in one diagram. We usually dispose source patterns on one side, and target patterns on the other side. Each pattern is layed in a graph manner, starting from the root stereotyped «pattern». Also, we use notes instead of other diagrams or UML artefacts in order to describe additional informations such as rule calls, constraints, small computation... We could imagine that such notes are generated from more appropriate UML constructs (activity diagrams, action language...), but for now we really want to be able to understand a rule in one glance.

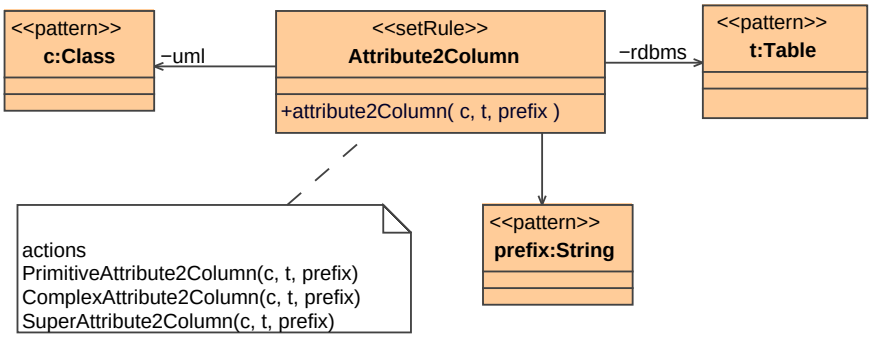

Figure 2: Description of the Attribute2Column rule