TROPIC – A Framework for Model Transformations on Petri Nets in Color[*]

[Extended Abstract]

M. Wimmer
TU Vienna
wimmer@big.tuwien.ac.at

A. Kusel
JKU Linz
kusel@bioinf.jku.at

J. Schoenboeck
TU Vienna
schoenboeck@bioinf.jku.at

W. Retschitzegger
University of Vienna
werner@bioinf.jku.at

W. Schwinger
JKU Linz
wieland.schwinger@jku.ac.at

ABSTRACT

Model transformation languages, the cornerstone of Model-Driven Engineering, often lack mechanisms for abstraction, reuse and debugging. We propose a model transformation framework providing different abstraction levels together with an extensible library of predefined transformations and a dedicated runtime model in terms of Coloured Petri Nets for transformation execution and debugging.

Keywords

Model Transformation, Mapping Operator, Reuse, Abstraction, Debugging, Colored Petri Nets, Runtime Model

1. INTRODUCTION

Model-Driven Engineering (MDE) places models as first-class artifacts throughout the software lifecycle, leading to a change from the "everything is an object" paradigm to the "everything is a model" paradigm [1]. The availability of proper model transformation languages is thereby *the* crucial factor, since transformation languages are for MDE as important as compilers are for high-level programming languages. Several kinds of dedicated model transformation languages have emerged (see e.g., [2] for a comparison), which allow, in a first phase, the *specification of transformations*, from elements of a source metamodel to elements of a target metamodel, and, in a second phase, the automatic *execution* thereof on the underlying models. None of these languages, however, not even the QVT standard proposed by the OMG, became generally accepted as state-of-the-art approach in practice, which seems to be, among others, due to the following reasons. Concerning the *specification phase*, firstly, existing model transformation languages do not provide appropriate abstraction mechanisms to deal with the complexity of overcoming structural heterogeneities between different metamodels, a form of heterogeneity well known in the area of database systems (cf., e.g., [5]), when specifying mappings between different schemata. Secondly, current approaches lack suitable reuse mechanisms in order to reduce the high and error-prone effort of specifying recurring transformations. Concerning the *execution phase*, firstly, transformation engines used for executing model transformations operate on a considerably lower level of abstraction than the specified mapping leading to an impedance mismatch between specification and execution. Secondly, current transformation languages provide a limited view on the execution of model transformations, since metamodels, models, transformation specification, and trace information are scattered across different artifacts, all of them hampering understandability and debuggabilty of model transformations.

[*]This work has been partly funded by the Austrian Science Fund (FWF) under grant P21374-N13.

2. GOALS

Our overall goal is to provide a framework for developing model transformations, in order to resolve structural heterogeneities between metamodels which tackles the aforementioned limitations of existing approaches. This overall goal can be further divided into three subgoals. Firstly, the specification phase should be supported by appropriate abstraction mechanisms and reuse facilities to increase productivity of transformation development and to ensure the quality of the resulting transformations. Secondly, the execution phase should be facilitated by a suitable representation of the runtime characteristics of a transformation together with debugging services to enhance understandability of transformations and to improve their correctness. Thirdly, the concepts developed for both phases should be applicable to certain selected existing model transformation languages such as the QVT standard.

3. APPROACH

To realize our goals, a dedicated framework called TROPIC (Transformations on Petri Nets in Color) is provided allowing the specification and usage of mapping operators for certain model transformation scenarios as well as the actual execution and debugging of transformations, either in a standalone manner or as a front-end for other model transformation languages. In particular, TROPIC provides two views on a transformation problem, namely an abstract *mapping view* which declaratively describes the semantic correspondences on a high level of abstraction and a *transformation*

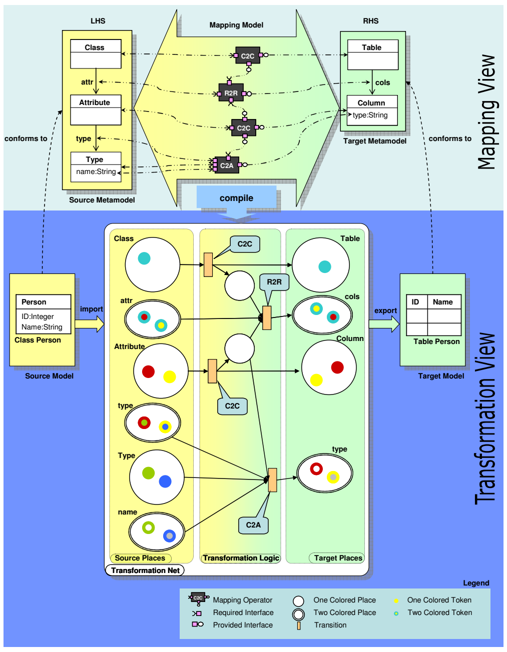

Figure 1: Two Views on a Transformation Problem

view which reveals all the details of the transformation logic being immediately executable (cf. Figure 1).

Mapping View. The mapping view comprises mapping operators which connect source metamodel elements to target metamodel elements [4]. These mapping operators *encapsulate recurring transformation logic* and are offered to a transformation designer by means of an *extensible library*. For defining the reusable mapping operators, we use a subset of the UML 2 component diagram concepts, since they are *declarative* in nature, provide a *black-box* mechanism to hide transformation logic details and support an interface concept allowing for the *composition* of mapping operators.

Transformation View. An executable transformation view is generated on basis of the mapping view. For this, each mapping operator of the mapping view must be provided with a well defined operational semantics in the form of some executable piece of transformation logic. For realizing the transformation view, we are using a modified form of Colored Petri Nets [3], denoted as Transformation Nets [6, 8, 9, 10] due to the following reasons. Firstly, Transformation Nets enable a process-oriented execution of transformations, each mapping operator being realized by an independent set of transitions and places without the need for specifying an explicit control flow, thus preventing any *impedance mismatch* between mapping view and transformation view. Secondly, Transformation Nets allow for a *homogenous representation of all artifacts involved* in a model transformation, thus being especially suited for gaining an understanding of the intricacies of a specific model transformation. Finally, Transformation Nets inherently form an *explicit runtime model* being immediately executable, thus facilitating the debugging of model transformations.

4. IMPLEMENTATION

A first prototype to build up the mapping view and the transformation view as well as to execute a certain transformation is already operational which will be applied in several case studies to verify our approach. This prototype is built on the Eclipse Modeling Framework (EMF) in combination with the Graphical Modeling Framework (GMF) and is realized as Eclipse plugin.

5. EVALUATION

In the course of evaluating our approach it should be tested, whether productivity, quality, understanding and debuggability of model transformations are increased through the application of metrics as e.g. proposed in [7]. For this purpose, we intend to conduct *case studies* with a representative selection of metamodels defining structural and behavioral languages as well as *empirical studies* with 200 master students of our MDE courses. Furthermore, we intend to arrange *collaborative studies* with three international project partners, being the inventors of other model transformation languages (Prof. Dr. Jean Bézivin, Prof. Dr. Andy Schürr) as well as of Colored Petri Nets (Prof. Dr. Kurt Jensen) in the form of dedicated workshops.

6. REFERENCES

[1] J. Bézivin. On the Unification Power of Models. *Journal on Software and Systems Modeling*, 4(2):171–188, 2005.

[2] K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.

[3] K. Jensen. *Coloured Petri nets: basic concepts, analysis methods, and practical use.* Springer, 1992.

[4] G. Kappel, H. Kargl, T. Reiter, W. Retschitzegger, W. Schwinger, M. Strommer, and M. Wimmer. A Framework for Building Mapping Operators Resolving Structural Heterogeneities. In *Proc. of Information Systems and e-Business Technologies (UNISCON'08)*, Klagenfurt, 2008. Springer.

[5] V. Kashyap and A. Sheth. Semantic and Schematic Similarities between Database Objects: A Context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.

[6] T. Reiter, M. Wimmer, and H. Kargl. Towards a runtime model based on Colored Petri Nets for the execution of model transformations. In *Proc. of 3rd Workshop on Models and Aspects @ ECOOP'07*, Berlin, 2007.

[7] M. van Amstel, C. Lange, and M. van den Brand. Using Metrics for Assessing the Quality of ASF+SDF Model Transformations. In *Proc. of ICMT2009 - Int. Conf. on Model Transformation Theory and Practice of Model Transformations*, Zurich, 2009. Springer.

[8] M. Wimmer, A. Kusel, T. Reiter, W. Retschitzegger, and W. Schwinger. Lost in Translation? Transformation Nets to the Rescue! In *Proc. of 8th Int. Conf. on Information Systems Technology and its Applications (UNISCON'09)*, Sydney, 2009. Springer.

[9] M. Wimmer, A. Kusel, J. Schoenboeck, G. Kappel, W. Retschitzegger, and W. Schwinger. Reviving QVT Relations: Model-based Debugging using Colored Petri Nets. In *MoDELS '09: Proc. of the 12th Int. Conf. on Model Driven Engineering Languages and Systems*, Denver, USA, 2009. Springer. Short Paper.

[10] M. Wimmer, A. Kusel, J. Schoenboeck, T. Reiter, W. Retschitzegger, and W. Schwinger. Let's Play the Token Game – Model Transformations Powered By Transformation Nets. In *Proc. of Int. Workshop on Petri Nets and Software Engineering*, Paris, 2009.