

MODEL TRANSFORMATION PLUGINS ON EMF MODELS

Ákos HORVÁTH

Advisor: Dániel VARRÓ

I. Introduction

Nowadays, model driven software development (MDSO [1]) is an emerging paradigm in software development. Based on high-level modeling standards (e.g. UML), MDSO separates business and application logic from underlying platform technology by using platform independent models (PIM) to capture the core functionality of the target system, and platform specific models (PSM) to specify the target system on the implementation platforms (Java, C#, C++). PSMs and platform-specific source code are automatically generated from PIM and PSMs, respectively, by using model transformation (MT) techniques, thus, the role of MT is unquestionable for the overall success of MDSO.

As the complexity of model transformations is growing, a new demand has been arisen to separate the design from transformation execution by using high-level models at design time and by automatically deriving source code for the target platform of the platform execution from these high-level models.

II. Model Transformation by Graph Transformation

Informally, a graph transformation [2] (GT) rule performs local manipulation on graph models by finding a matching of the pattern prescribed by its left-hand side (LHS) graph in the model, and changing it according to the right-hand side (RHS) graph. GT rules are capable of manipulating the model locally, but every MT concept which is based on GT rules connects additional control structure to support complex MT constructs, so more and more complex MTs can be captured by a sequence of GT rules. But regardless of the approach of the model transformation, these concepts should enable a cost and time efficient design of (i) manipulations within a single modeling language (or domain) (ii) mappings and synchronization between different modeling languages and/or source code (iii) semantic translations into various mathematical domains to carry out formal model analysis.

III. Eclipse Modeling Framework

Eclipse Modeling Framework (EMF) [3] is currently one of the defacto standards for modeling used by leading software companies (e.g., IBM, SUN, etc.) and supported by the Eclipse foundation. EMF is a Java framework and code generation facility for building tools and other applications based on a structured model. EMF provides a metamodel for describing structured models. Using these structured models EMF provides tools and runtime support to produce a set of Java classes representing the model in Java, a set of adapter classes that enable viewing and a basic editor.

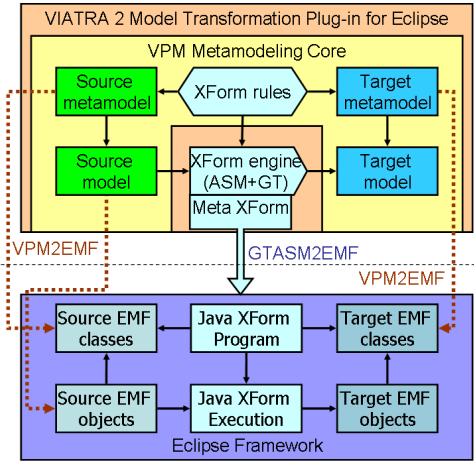
The ECore framework is used to create the metamodels. ECore is essentially the class diagram subset of UML. It is based on the Object Management Groups (OMG) Meta Object Facility (MOF) specification. From an ECore model, the generator of EMF can create a corresponding set of Java implementation classes. Every generated EMF class extends from the framework base class, EObject, which enables the objects to be integrated and appear in the EMF runtime environment.

IV. Overview of the Approach

The current paper describes a new approach using high order (meta [4]) transformation rules for the source code generation from high-level model transformation specifications defined by a combination

of graph transformations and abstract state machine (ASM) constructs (as used within the VIATRA2 [5] framework, a general Eclipse based modeling framework).

The essence of the approach is to store model transformation rules (XForm rules) as ordinary models in the model space. This way the source code generator (Meta XForm) of the transformations can be implemented within the modeling framework. As a result, the code generator can be reused by replacing only the output generation rules in order to port the transformations to new execution platforms.



The proposed solution consists of two main transformations. The *VPM2EMF* transformation is responsible for mapping the models and metamodels described in VIATRA2 into EMF models. The derivation contains two steps: (i) the Source and Target metamodels are mapped to Source and Target EMF classes, (ii) the generated Java classes representing the ECore model are instantiated based on the source model within the VIATRA2 framework.

Now that the source and target EMF classe are derived from the VIATRA2 frame-

work, the *GTASM2EMF* transformation maps the high-level transformation specification of the framework into native Java code. Because of the joint representation of models and transformations this can also be done within VIATRA2 using meta-transformations (i.e., a transformation having transformations as its input and output). The model manipulation operations of the specification are mapped to reflective API calls provided by the EObject.

This approach enables not only to run transformations as standalone application but also to integrate them with other java (Eclipse) based applications, which in case of specific tasks can significantly shorten development time (e.g., XML-to-XML mapping).

V. Conclusion

In the current paper, we proposed to meta-transformations for generating EMF-based transformer plugins from transformation specifications given by the combination of graph transformation rules and abstract state machines in the VIATRA2 framework. The completion of the implementation and the optimization of the model manipulation is an ongoing research.

In the future we would like to use this generator as a base platform for advanced transformation verification such as Proof Carrying Code [6] (PCC).

References

- [1] J. Bettin, "Model-driven software development," URL: <http://www.softmetaware.com/mdsd-and-isad.pdf>.
- [2] G. Rozenberg, Ed., *Handbook of Graph Grammars and Computing by Graph Transformation, volume 1: Foundations*, World Scientific, 1997.
- [3] "Eclipse Modeling Framework," URL: <http://www.eclipse.org/emf>.
- [4] D. Varró and A. Pataricza, "Generic and meta-transformations for model transformation engineering," in *Proc. UML 2004: 7th International Conference on the Unified Modeling Language*, T. Baar, A. Strohmeier, A. Moreira, and S. Mellor, Eds., vol. 3273 of *LNCS*, pp. 290–304, Lisbon, Portugal, October 10–15 2004. Springer.
- [5] *VIATRA2 Framework*, An Eclipse GMT Subproject, URL: <http://www.eclipse.org/gmt/>.
- [6] G. C. Necula, "Proof-carrying code," in *Proc. POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 106–119, Paris, France, 15–17 1997.