Model-Driven Development of Model Transformations

Pieter Van Gorp

University of Antwerp
pieter.vangorp@ua.ac.be

Abstract. The "model-driven development of model transformations" requires both a technique to *model* model transformations as well as a means to *transform* transformation models. Therefore, the thesis underlying this paper evaluates and extends state-of-the-art model transformation approaches. For example, the thesis contributes a new language construct for modeling subgraph-copy operations. Perhaps surprisingly, this thesis intentionally does *not* propose a fundamentally new transformation language and toolset. Instead, the thesis is based on a small UML profile for controlled graph transformation. The profile only relies on class diagrams, activity diagrams, and the UML's extension mechanism. The proposed techniques have emerged from several case studies that involve model *evolution*, model *refinement*, as well as model *synchronization*.

1 Problem: Lack of Portability and Reuse

Controlled graph transformation gained industrial credibility in the nineties, thanks to the application of the Progres language (and tool) within industrial tool integration projects. After working within the Progres team, so-called "Story Diagrams" were proposed as a UML based syntax for modeling graph transformation systems. Since some implementation challenges were hard to overcome on top of the C based implementation of the Progres tool, the Java based Fujaba tool was implemented. With the advent of the MDA, several comparable tools were constructed once more. Unfortunately, several tools have proposed yet another syntax for existing graph transformation constructs. Moreover, none of the graph transformation tools relied on common libraries to reuse existing infrastructure for pattern matching, control flow, etc.

2 Solution: A standard Transformation Modeling Profile

The first step to transformation tool integration is the agreement on a common metamodel for representing transformation models. Whenever possible, such a metamodel should be aligned with mainstream standards. As stated in the previous section, "Story Diagrams" was the first language for representing controlled graph transformation models in mainstream UML syntax. Unfortunately, the language was initially based on a proprietary and rather implicit metamodel.

Therefore, as a first contribution to the thesis, we aligned that controlled graph transformation language with the UML metamodel. Several off-the-shelf UML tools can now be used to edit the transformation models based on Story Diagrams. Moreover, the tool that supports this thesis relies on another mainstream MDA tool to transform transformation models into standard (i.e., MOF/JMI) compliant repository code. This lowers the cost and risk for adopting the proposed approach in an industrial context. Unlike the emerging QVT standard, there is no need to learn a completely new language and buy into a completely new toolset.

The core profile for modeling model transformations only supports the basic concepts of controlled graph transformation: it has the notion of a rewrite rule (matched elements, created elements, deleted elements and updated elements) and control flows (iterative loops, conditionals and called transformations). Interestingly, these constructs already enable one to model refactorings [3] as well as refinements [5] in a human friendly manner. Nevertheless, the proposed approach enables one to add more expressive language constructs instead of prematurely standardizing all transformation tools to the bare minimum of their "lowest common denominator".

3 Extensibility: Higher Order Transformations

The previous section only indicated how transformation model editors can be integrated at the syntactic level. The thesis proposes to define new language constructs as extensions to a small (the "core") transformation modeling profile. The role of higher order transformations is to transform transformation models that conform to an extension of the profile into transformation models that conform to the core profile (of which the semantics has been standardized).

A key to the proposed approach is that the higher order transformations themselves are modeled using the core profile for transformation modeling [4, Chapter 8]. Therefore, any transformation engine that supports the core profile can execute the higher order transformations. Consequently, any such engine can normalize transformation models that apply a new language construct into more primitive transformation models. Today, only a *Copy* operator has been realized using this approach. However, as new operators (such as *Merge*, *Diff*, ...) are introduced, a transformation tool may need to execute a series of publicly available higher order transformations before executing the result on its native graph transformation engine.

4 Related Work

First of all, Graph Transformation eXchange Language (GTXL [2]) has been proposed as a standard for exchanging transformation models. Unlike the proposed profile, GTXL has no relation to a mainstream modeling language such as the UML. Therefore, there are no off-the-shelf industrial tools for editing GTXL models. Secondly, the GTXL metamodel relies on a XML DTD instead of on the MOF. Therefore, it requires more integration effort in an MDA tool integration context. Finally, GTXL only supports uncontrolled rules whereas the proposed profile supports rules that are controlled by activity diagrams. Finally, due to the lack of a profile concept, GTXL cannot be extended without breaking metamodel compatibility with its implementations.

Secondly, the Queries/Views/Transformations (QVT) standard presents three languages for transformation modeling. Apart from the MOF basis, it has the same limitations as GTXL. The QVT standard does promote bridges between its sublanguages by means of higher order transformations. In fact, the mapping between the relations and core language is formalized in the QVT relations language. Unfortunately, the QVT relations language is not as generally applicable as the profile presented in this paper.

Within the VIATRA tool, the transformation process from human-oriented transformation models into machine-oriented transformation code is supported by higher order transformations too [1]. Unlike the proposed approach, the transformation models do not conform to any standards. Moreover, the higher order transformation is not written in a standard transformation language either.

5 Conclusions

This paper presented a new approach to the integration and extension of transformation languages and tools. A realization of the proposed approach enables transformation tool builders to focus on user-oriented added value (such as editor usability, run-time performance, ...) and new, *declarative* language constructs (such as a *Copy* operator), instead of spending time on the implementation of evaluation code that was already been realized in other tools before. A unique characteristic of the approach is that it only requires transformation tool builders to implement a small core, and provides support for more declarative language constructs without breaking interoperability.

References

1. Ákos Horváth, Dániel Varró, and Gergely Varró. Automatic generation of platform-specific transformation. *Info-Communications-Technology*, LXI(7):40–45, 2006.
2. Leen Lambers. A new version of GTXL : An exchange format for graph transformation systems. *Electronic Notes in Theoretical Computer Science*, 127:51–63, March 2005.
3. Hans Schippers, Pieter Van Gorp, and Dirk Janssens. Leveraging UML profiles to generate plugins from visual model transformations. *Electronic Notes in Theoretical Computer Science*, 127(3):5–16, 2004. Software Evolution through Transformations (SETra). Satellite of the 2nd Intl. Conference on Graph Transformation.
4. Pieter Van Gorp. *Model-driven Development of Model Transformations.* PhD thesis, University of Antwerp, April 2008.
5. Pieter Van Gorp, Olaf Muliawan, Anne Keller, and Dirk Janssens. Executing a platform independent model of the UML-to-CSP transformation on a commercial platform. In Gabriele Täntzer and Arend Rensink, editors, *AGTIVE 2007 Tool Contest*, January 2008.