Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 133

Case Studies for Bidirectional Transformations in QVT Relations

Bernhard Westfechtel[1]

Abstract: QVT Relations (QVT-R), a standard issued by the Object Management Group (OMG), is a language for the declarative specification of model transformations. In particular, QVT-R supports the specification of bidirectional transformations: Rather than writing two unidirectional transformations separately, a transformation developer may provide a single relational specification which may be executed in both directions (from source to target and vice versa). In this contribution, which is based on [We16], we summarize the main results from a series of case studies which shed a light on both potentials and limitations of QVT-R as a bidirectional transformation language.

Keywords: Model-driven software engineering, bidirectional transformations, QVT Relations

1 Background

Model-driven software engineering (MDSE) promises to improve the productivity of software engineers by developing software systems from high-level models from which executable code is derived through a series of model transformations. To support MDSE, languages for expressing both models and model transformations are needed. For object-oriented modeling, the Object Management Group (OMG) has defined a series of standards, including e.g. UML as a modeling language and MOF as a metamodeling language. In addition, the OMG provides the QVT standard (queries, views, and transformations [QVT15]), which defines a family of MOF-based transformation languages.

In particular, the declarative language QVT Relational (QVT-R) allows to specify transformations in terms of relations between patterns in source and target models. A transformation written in QVT-R may be uni- or bidirectional and may be executed in two modes: In checking mode, it is merely checked whether the target model is consistent with the source model. In enforcing mode, the target model is updated incrementally to enforce consistency. Batch transformations are treated as a special case (empty target model).

This paper focuses on QVT-R's support for bidirectional transformations, which are required in a variety of use cases (e.g., bidirectional data converters or round-trip engineering between models and code). In QVT-R, a single transformation definition may be executed in up to six different ways (checks, batch and incremental transformations, both in forward and backward direction). This approach promises to reduce the specification effort significantly; in addition, it assists in ensuring the consistency between forward and backward transformations (ideally, the transformations invert each other).

[1] Universität Bayreuth, Lehrstuhl für Angewandte Informatik 1, Universitätsstr. 30, 95440 Bayreuth, bernhard.westfechtel@uni-bayreuth.de

2 Goals and Approach

We investigated QVT-R's support for bidirectional transformations through a series of cases from the perspective of a transformation developer. In [We16], we studied seven cases which vary according to different factors (e.g., size, restructuring, and information loss), developed solutions in QVT-R and evaluated them with respect to different features such as conciseness, expressiveness, cognitive complexity and semantic soundness. In addition to the synthesized cases presented in [We16], we explored a real case from our research tool (Valkyrie) for UML. [SGW16] presents a QVT-R specification for the model-to-code roundtrip (UML class diagrams $\leftrightarrow$ Java code). This solution was compared in [BG16] against a hand-crafted solution in an object-oriented programming language.

3 Results

Most cases were solved with a bidirectional tranformation definition exhibiting the expected behavior. Surprisingly, we failed in developing bidirectional transformation definitions for two cases with bijective mappings, where the representation of information is merely restructured (e.g., transformation between expression trees and dags). In some cases, development of the bidirectional transformation was more difficult than expected. While QVT-R is a declarative language, the transformation developer needs to consider the execution behavior carefully to ensure that the transformation actually works as expected in each mode and direction. This task is complicated further by the fact that the semantics is not defined in a sound way in the standard and that support tools do not even try to conform to the semantics definition in the standard. Finally, in some cases we observed redundancies in the rule sets (many rules with different functionalities). Thus, the size of a transformation definition may be even larger than the size of a procedural solution (for model-to-code, more than twice as many lines of code than the hand-crafted solution [BG16]).

References

[BG16] Buchmann, Thomas; Greiner, Sandra: Handcrafting a Triple Graph Transformation System to Realize Round-trip Engineering Between UML Class Models and Java Source Code. In (Maciaszek, Leszek; Cardoso, Jorge; Ludwig, André; van Sinderen, Marten; Cabello, Enrique, eds): Proceedings of the 11th International Conference on Software Paradigm Trends. SCITEPRESS, pp. 27–38, July 2016.

[QVT15] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.2. Needham, MA, formal/2015-02-01 edition, February 2015.

[SGW16] Sandra Greiner, Thomas Buchmann; Westfechtel, Bernhard: Bidirectional Transformations With QVT-R: A Case Study in Round-trip Engineering UML Class Models and Java Source Code. In (Selic, Bran; Desfray, Philippe, eds): Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016). SCITEPRESS, p. 13, Feb 2016.

[We16] Westfechtel, Bernhard: Case-based exploration of bidirectional transformations in QVT Relations. Software and Systems Modeling, May 2016. Online First.