Suitability of Active Rules for Model Transformation

(Demo paper)

Lingzhe Liu* and Manfred A. Jeusfeld+

* lliu@rsm.nl, Erasmus University, Rotterdam, The Netherlands

+ manfred.jeusfeld@acm.org, Tilburg University, The Netherlands

Abstract. Model transformation promises faster and higher-quality system development processes by automating certain development steps. There are numerous proposals such as QVT and triple graph grammars that are applied in practical design environments. To our surprise, active rule systems have not yet been considered as a platform to execute model transformations. We investigate in this paper the suitability of active rule systems via a case study. An empirical analysis of the complexity is provided as well. It turns out that active rules support the basic functional requirements but some extensions to their execution engine and join order optimization would be needed.

Keywords. Model-driven architecture, model transformation, active rule

1 Introduction

The model-driven architecture aims at lifting software development to a higher, more abstract level [KWB2003]. If mappings exists that translate from a more abstract model towards a more concrete one, then the effort would be shifted from coding towards modeling.

To realize the vision, models for all abstractions levels (specification, design, implementation) are represented as instances of meta models, in particular MOF-based. The state of the art of model transformation is materialized in tools based on QVT, and on triple graph grammars (TGGs). Surprisingly, there was so far no attempt to use active rules for the task of model transformation. The models can be represented in an active database, and active rules can encode the transformations between models. In particular, fine-grained incremental updates can be supported.

In this paper, we investigate in more detail the suitability of active rules for the task model transformation. In the subsequent chapter, we study the state of the art in model transformation and establish the requirements that a solution based on active rules should fulfill. Afterwards, a case study on realizing a UML-to-Relational-Database mapping is presented.

2 State of the Art

Model transformation has a language aspect and a tool aspect [CH2003,CH2006]. The *transformation language* encodes the specification of the model transformation, i.e. what elements should be transformed. The *transformation tool* provides the engine interpreting the specification, i.e. realizes how the transformation is performed.

Model-to-code transformation can be regarded as a special case of *model-to-model* transformations, since models can be explicated as a linear textual representation. *Declarative* approaches are contrasted to *imperative* approaches. The latter describe the steps of transforming source models to target models, while the former describe the relations between elements of the source and target models. Imperative approaches hence amalgamate the tooling aspect with the language aspect. We pick two declarative transformation languages to extract concrete requirements for model transformation: QVT (query-view-transformation) and TGG (triple graph grammars). Both languages are rule-oriented, .i.e. the transformation is specified by a set of transformation rules.

2.1 QVT-Core

QVT [QVT2009,Ecl2011,JK2006,XLH*2007,Kur2008] comes in three flavors, QVT-Relations (defining transformations as a set of relations), QVT-Core (defining the semantics of QVT-Relations with a simpler set of language construct), and QVT-Operational (imperative flavor of QVT). We concentrate subsequently on QVT-Core.

QVT-Core is *multi-directional* in nature. The same rule can be read in both directions, but only if the underlying logic of the transformation is the same. Modularity is supported by defining modules containing transformations, which themselves contain the individual mappings. The smaller parts inherit context settings from the larger parts. A transformation rule in QVT-Core distinguishes three areas: the left hand side (source model), the right-hand side (target model), the middle area (traceability objects represented as ordinary model elements). The latter maintain the dependencies between the generated elements of the target model and the elements of the source model(s). The transformation rule makes a test (“guard pattern”) on all three areas, checking which elements exists, and demands in its “bottom pattern”, which elements in the target and middle areas shall be generated for a given element on the source side (“realized variables”). Variables are bound to elements that may stem from different models.

QVT-Core can specialize transformation rules via a refinement feature. It inherits all mappings from the refines rule that are not overruled or removed. Moreover, there is a nesting mechanism which binds variables at the higher level that are then used at the nested levels.

2.2 TGG

Triple-graph grammars [KS2006] extend graph grammars by a middle graph that basically represents the traceability objects between a left-hand side (LHS) and the right hand side (RHS). Both the LHS and the RHS state dependencies between elements of the source model(s) and elements of the target model(s). The LHS represents the current state of the transformation, i.e. the pre-condition. The RHS declares which elements are present after application of the rule, adding new traceability objects and new objects in the target model, and possibly also in the source model. Besides the test on (non-) existence of nodes and links, TGGs also support cardinality pattern, e.g., that a node has exactly n others nodes linked to it.

Since TGGs are by nature non-deterministic, the actual decision on rule execution is done by the transformation tool [Agra2003]. Round-trip transformation with graph grammars is supported by the Fujuba tool [GZ2005].

3 Case Study: UML-RDBMS

The complexity of the modeling language leads to complex specifications of the model transformation. The purpose of this section is to find out whether ECA rules scale for realistic model transformations, both in terms of specification complexity (here: size of the specification) and the execution time. We use the ECA rule implementation of ConceptBase for both purposes. It is in principle Turing-complete and allows to represent models of many different modeling languages due to its metamodeling facility.

The QVT-Core example transformation UML-RDBMS [OMG 2009] is the benchmark transformation. It consists of 366 lines of QVT-Core code to transform a (simplified) UML class diagram to a relational schema including primary and foreign key specifications. As QVT also employs mapping objects, they form the basis for defining the ECA rules[1]. For example, consider the QVT rule `packageToSchema`:

```
map packageToSchema in umlRdbms {
        uml ()   { p:Package }
        rdbms () { s:Schema  }
        where () {
            p2s:PackageToSchema|
            p2s.umlPackage = p;
```

1 A detailed presentation of the representation of the QVT-style mapping with ECA rules is given in [Liu2010]. The full example including all ECA rules is online at http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/3015942.

```
            p2s.schema = s; }
        map { where () {
              p2s.name := p.name;
              p2s.name := s.name;
              p.name := p2s.name;
              s.name := p2s.name; } } }
```

Its representation as an ECA rules for the direction towards RDMS is:

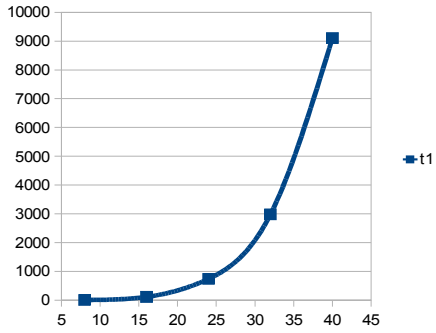
```
UnmatchedPackage in QueryClass isA UPackage with
  constraint
    c1: $ exists name1/String
    (~this name name1) and
    (not (~this matchable FALSE)) and
        (not exists p2s1/PackageToSchema
           (p2s1 umlPackage ~this) and
           (p2s1 name name1)
        )$
end

create_p2s_tr_s in ECArule with
  mode m: Deferred
  inTrans intrans : umlRdbms
  nonMappingRuleFlag mf: FALSE
  exedirn exedir2: SimpleRdbms {* Mapping Direction *}
  ecarule
    mr_p2s_s : $ p/UPackage
                 name1/String p2s1/PackageToSchema
                 tr/Transformation
    ON Ask exeTrans[tr/trans] {* top level mapping *}
    IF NEW (p in UPackage) and
       (p in UnmatchedPackage) and
       (p name name1)
    DO CALL CreateIndividual(P2S,p2s1),
       Tell (p2s1 in PackageToSchema),
       Tell (p2s1 name name1),
       Tell (p2s1 umlPackage p)
    $
end
```

The first clause `UnmatchedPackage` defines an auxiliary query checking whether the source model has not yet been mapped. The ECA rule `create_p2s_tr_s` uses the tag '`IF NEW`' to indicate that the condition is tested against the new database state, i.e. including the updates done by previous ECA rule executions. The translation of the QVT-Core specification UML-RDBMS to ECA rules required 1504 lines of code. This includes about 400 lines of code for the specification of the meta models of UML class diagrams and RDBMS. It should be noted that the QVT-Core rules are bi-directional. Hence, at least two ECA rules had to be coded per QVT-Core rule. Still, the ECA rule coding is about 4 times longer. Of the 34 ECA rules, 16 are in mode 'Deferred' and 18 in mode 'Immediate'. Additionally, 38 query classes are defined to

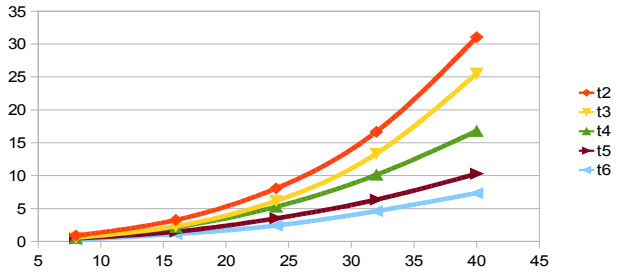
check the structure of the source model and the current state of the model transformation. In combination, the ECA rules support both mapping directions.

To check the performance of the ECA rule representation, we created five UML class diagrams with 8, 16, 24, 32, and 40 types (classes and associations) and measured the transformation times. In the first run t1 (see chart on the right), the conditions of the ECA rules where evaluated in the order in which they occurred.



The X- axis represents the input model size, the Y-axis is the transformation time in seconds. This is almost an intractable performance with a polynomial regression function close to $O(n^4)$. The reason is the lack of query optimization on the conditions of the ECA rules. Hence, we included several optimizations strategies in the ECA system of ConceptBase leading to the following execution times:

The best variant t6 is almost linear and orders of magnitude better than variant t1! It has a small quadratic factor that is due to the 'IF NEW' construct.



ConceptBase uses tabling for evaluating derived predicates and queries. The tabled extensions of the predicates speed up the computation. However, when the condition is evaluated against the new database state, then the old extension is no longer valid and has to be re-computed. This re-computation happens for each ECA rule execution. One can expect that an incremental update of the tabled predicate extensions would remove the small quadratic factor from the transformation time. The transformation of an input model of size 40 requires about 7.5 seconds on a 2.4 GHz CPU. A considerable portion is due to the relatively expensive Tell operation of ConceptBase. It maintains several indexes to store facts and each Tell includes transformations from names to identifiers, and from identifiers to memory addresses.

Table 1: Comparison of the three approaches

Category	Criteria	Transformation Approach		
		QVT-Core	**TGGs**	**ECA rules**
Syntax & Expressiveness (Language aspect)	scoping	guard patterns	LHS	On-part and IF-part
	pattern for source domains	bottom patterns of source domains	LHS	IF-part
	pattern for checking target domains	bottom patterns of target domains	RHS (read w/ attribute constraints)	IF-part, QueryClass
	pattern for enforcing target domains	bottom patterns of target domains	RHS (write w/ attribute assignments)	DO-part
	specification of constraint and assignment	logical spec., assignments as constraint in check mode	graphical spec., w/ simple constraints and assignments	IF-part, assignments in DO-part
	directions	bi-directional	bi-directional	unidirectional
	modularity	module, transformation and map	N/A	limited, meta model
	reuse	refinement of mappings	reusable node (in Fujaba)	reuse of post-condition
	composition	nested mapping	graphic composition	reuse of post-condition
	n:1(wrt. elements)	supported	supported	supported
	n:m(wrt. elements)	supported	supported	simulate
	N:1 (wrt. models)	supported	supported	meta model
	N:M(wrt. models)	simulate	supported	meta model
	logical constraints	supported	partially supported	supported
Execution semantics (Tooling aspect)	execution condition	N/A, mappings with valid matches are always executed	block (in [AKS2003])	1. IF-part 2. `active` attribute of `ECArule`
	location determinism	non-deterministic, need tool support	non-deterministic, need tool support	deterministic, pattern matching begins with the triggering element

4 Conclusions

The purpose of this paper was not to present yet another model transformation approach. We rather explored the suitability of existing active rule systems to *implement* model transformations. This was not investigated before. We argue that active rules are a quite natural platform for model transformation. The main result is that active rules are suitable for the task. The coding overhead is manageable, and there is practically no performance penalty. From the viewpoint of active rules, the modularity and bi-directionality are shortcomings that need to be addressed. We believe that modularity can be defined with a suitable meta model. Bi-directionality requires a code generation approach, e.g. generating ECA rules from a QVT-Core specification. This is in close reach as the case study indicates.

Active rules provide more expressive power than the other approaches, e.g. for analyzing the source/target models and the current state of the transformation. The generation of ECA rules from the more abstract QVT or TGG specification is subject to future work. Further research shall also focus on properties like termination and confluence of the ECA rules, and the use of metrics in mapping rules.

References

[AH1988] S. Abiteboul, R. Hull. Data Functions, Datalog and Negation (Extended Abstract). Proc. ACM SIGMOD, Chicago, USA, 143-153.

[AKS2003] A. Agrawal, G. Karsai, G., F. Shi. Graph Transformations on Domain-Specific Models. Technical Report, Vanderbilt Univ., Nov. 2003.

[AWH1995] A. Aiken, J. Widom, M. Hellerstein. Static analysis techniques for predicting the behavior of active database rules. *ACM TODS* 20(1):3-4, 1995.

[AFN2010] M. Arenas, R. Fagin, A. Nash. Composition with target constraints. Proc. ICDT'2010, Lausanne, Switzerland, March 20-22, 2010, 129-142.

[CLST2010] D. Calegari, C. Luna, N. Szasz, A. Tasistro. A Type-Theoretic Framework for Certified Model Transformations. Proceedings of SBMF'2010, Springer LNCS 6527, 112-127, 2010.

[CH2003] S. Czarnecki, S. Helsen. Classification of Model Transformation Approaches OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture.

[CH2006] S. Czarnecki, S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45(3):621-645, 2006.

[DGG1995] K.R. Dittrich, S. Gatziu, A. Geppert: The Active Database Management System Manifesto: A Rulebase of ADBMS Features. Springer, *LNCS* 985, Springer 1995, ISBN 3-540-60365-4, 3-20.

[Ecl2011] Eclipse Foundation. Model 2 Model (M2M). Online http://www.eclipse.org/m2m/.

[GZ2005] H. Giese, A. Zündorf (eds.). Fujuba Days 2005. Proceedings, Paderborn, Germany, 2005.

[KWB2003] A. Kleppe, J. Warmer, W. Bast. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley:Boston, 2003.

[Jeu2009] M.A, Jeusfeld. Metamodeling and method engineering with ConceptBase. In Jeusfeld, M.A., Jarke, M., Mylopoulos, J. (eds): Metamodeling for Method Engineering, 89-168. The MIT Press, 2009.

[JK2006] F. Jouault, I. Kurtev. On the architectural alignment of ATL and QVT. Proceedings ACM Symposium on Applied Computing, Dijon, France, April 23-27, 2006, 1188-1195.

[Kur2008] I. Kurtev. State of the Art of QVT : A Model Transformation Language Standard. *Data Engineering* 5088(ii): 377-393.

[Liu2010] L. Liu. Active Rules for Model Transformation. Master thesis. Tilburg University, The Netherlands, 2010.

[Nic1982] J.-M. Nicolas. Logic for Improving Integrity Checking in Relational Data Bases. Acta Inf. 18:227-253, 1982.

[QVT2009] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.1, Object Management Group, 2009.

[XLH*2007] Y. Xiong, D. Liu, Z. Hu, H. Zhao, M. Takeichi, H. Mei. Towards automatic model synchronization from model transformations. Proceedings 22nd IEEE/ACM Intl. Conf. on Automated Software Engineering., 2007.

- - -

This paper is a companion paper for the demonstration of the active-rule based model transformation at the CAiSE 2012 Forum. The active rules of the case study are available from

http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/3015942

They can be executed with the ConceptBase system available from

http://conceptbase.cc.