TROPIC - A Framework for Building Reusable Transformation Components*

Angelika Kusel

Information Systems Group
Johannes Kepler University Linz
Altenberger Straße 69, 4040 Linz, Austria
kusel@bioinf.jku.at

Abstract. Model transformation languages are crucial for the success of Model-Driven Engineering (MDE), being comparable to the importance of compilers for high-level programming languages. The support of large transformation scenarios, however, is still in its infancy since the development of transformations currently takes place on a low-level of abstraction, lacking appropriate reuse mechanisms. We propose a framework called TROPIC (Transformations on Petri Nets in Color) for developing model transformations which tackles these limitations. Firstly, TROPIC allows to specify model transformations on different abstraction levels by providing an abstract mapping view and a concrete transformation view. Secondly, TROPIC facilitates reusability by providing an extensible library of reusable transformation components leading to increased productivity of model transformation development and to higher quality of the resulting model transformations.

Key words: Generic Model Transformations, Reuse, Abstraction

1 Introduction and Problem Description

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]. In this respect, model transformations play a vital role, representing *the* key mechanism for *vertical transformations* like the generation of code or documentations and *horizontal transformations* like translations, augmentations and alignments of models, to mention just a few. Several kinds of dedicated model transformation languages have emerged (see [2] for a comparison), which allow specifying and executing transformations between source and target metamodels and their corresponding models, respectively. None of these languages, however, not even the QVT-standard [3] proposed by the OMG, became generally accepted as a state-of-the-art approach. This rare adoption of model transformation languages in practice seems to be, among others, due to the following reasons. 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

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

systems when creating mappings between different schemata [4]. Secondly, current approaches lack *suitable reuse mechanisms* in order to reduce the high and error-prone effort of specifying recurring transformations.

2 Proposed Solution

To alleviate the above mentioned problems, a framework for building reusable transformation components is proposed (denoted as *mapping operators* in the following) which are used to resolve recurring transformation problems in model translation scenarios (cf. Figure 1 (a)). The framework 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 view* which reveals all details of the transformation logic.

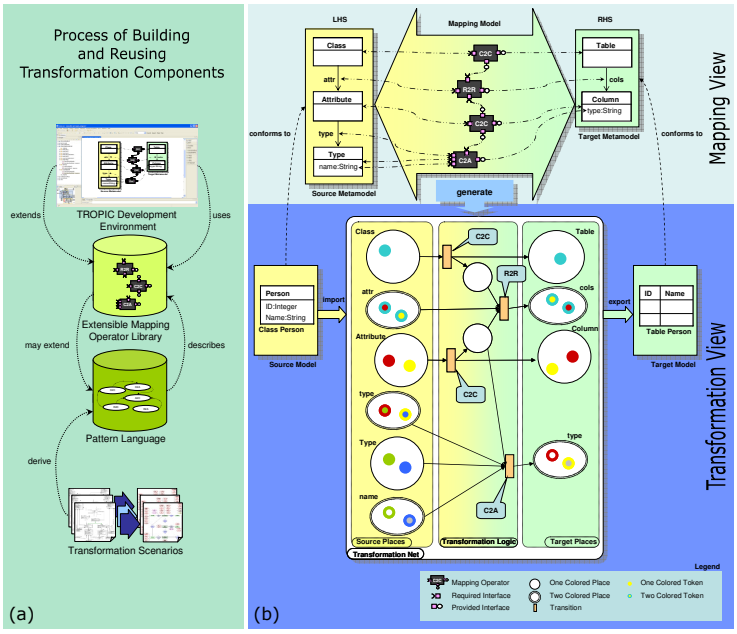

Fig. 1. (a) Mapping Framework (b) Multiple Views on a Transformation Problem

Mapping View. The mapping view level comprises mapping operators which connect source metamodel elements to target metamodel elements. These mapping operators encapsulate recurring transformation logic and are offered to a transformation designer by means of an extensible library. As a representation formalism, we intend to use a subset of the UML 2 component diagram concepts due to the following reasons. Firstly, this formalism supports a declarative description of mappings. Secondly, a black-box view for transformation logic is provided. And finally, the component's provided and required interfaces enable the composition of mapping operators in order to resolve complex structural heterogeneities. These interfaces are typed by the meta-metamodel datatypes

(i.e. in Ecore EClass, EReference and EAttribute), allowing mapping operators to be bound to arbitrary metamodels.

In order to exemplify our approach, Figure 1 (b) illustrates the overall idea from a user's point of view. For this, a simple example is used, transforming some basic object-oriented concepts (`Classes`, `Attributes` and `Types`) into corresponding relational concepts (`Tables` and `Columns`). In order to resolve the occurring structural heterogeneities, three different mapping operators are used, namely a *C2C-component* (transforming class instances, e.g., of the class `Attribute` into instances of the class `Column`), a *R2R-component* (transforming reference instances, e.g., of the reference `attr` into instances of the reference `cols`) and a *C2A-component* (transforming class instances into attribute instances, e.g., of the class `Type` into instances of the attribute `type`).

Transformation View. On basis of this mapping view, an executable transformation view is generated. For this, each mapping operator of the mapping view must have a well-defined operational semantics in the form of some executable piece of transformation logic. For realizing the transformation view, we are planning to use a modified form of Coloured Petri Nets [5], in the following denoted as Transformation Nets [6] due to the following reasons. Firstly, Transformation Nets enable the execution of the transformation without introducing an *impedance mismatch between the mapping view and the transformation view* as each mapping operator can be realized by an independent set of transitions and places without the need for an explicit control flow between the mapping operators. Secondly, this formalism allows 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, since Transformation Nets are already executable, an *explicit runtime model* is provided facilitating the debugging of model transformations [7].

3 Expected Contributions

Three main contributions are expected which foster reuse and abstraction allowing for larger transformation scenarios. Firstly, abstract reuse will be supported by the development of a *pattern language* for model transformations. Subsequently, concrete reuse will be supported by offering an extensible mapping operator library. Finally, abstraction will be facilitated through a development environment, that can be used to realize a mapping view on a concrete transformation problem and generate the corresponding exeutable transformation logic.

Pattern Language for Model Transformations. A major task will be the investigation of existing model transformations to build up a catalog of transformation patterns for recurring transformation problems in the form of a textual description comprising the standard parts of a design pattern, i.e., name, description as well as concrete implementation. For identifying these patterns, different sources will be investigated like (1) existing lists of patterns for resolving structural heterogeneities [8], [9], (2) existing model transformations in the ATL model transformation zoo (www.eclipse.org/m2m/atl/atlTransformations/), and (3) transformation scenarios between metamodels for

structural domains (e.g., ER models and UML class models) as well as for behavioral domains (e.g., BPMN models and BPEL models). Finally, common problems in the area of information integration will be investigated since the mapping of schemas is closely related to the mapping of metamodels whereby, [10] and [11] provide starting points. On top of the resulting list of found patterns, a useful categorization will be established resulting in a pattern language.

Extensible Mapping Operator Library. It goes without saying, that the resulting library of mapping operators being part of the pattern language can not be complete with regard to solving arbitrary transformation problems. Therefore, the transformation designer must be able to define his/her own mapping operators leading to the need of a mapping operator editor which allows to extend the library of existing mapping operators by user-defined ones and thus potentially extending the pattern language. User-defined mapping operators can be defined from scratch or by reusing existing ones. In this respect, different reuse mechanisms should be possible like building a new operator by (1) *black-box reuse* comprising the sequencing and/or nesting of existing ones or by (2) *white-box reuse*, i.e. inheriting from an existing one and further refining it.

Development Environment. Finally, mapping operators must be applicable in concrete model transformation scenarios representing the mapping view of a transformation problem. Therefore, a development environment is needed, which allows first, to build a mapping model consisting of mapping operators between a concrete source metamodel and a concrete target metamodel and second, to generate the corresponding executable transformation view.

4 Related Work

Related Work is discussed along three dimensions: abstraction, abstract reuse and concrete reuse.

Abstraction. The ATLAS Model Weaver (AMW) [12] offers abstraction mechanisms by the definition of simple correspondences (denoted as weaving operators) between two metamodels. The operational semantics of the weaving operators is determined by a higher-order transformation that takes a weaving model as input and generates model transformation code. The weaving models are compiled into low-level transformation code in terms of ATL which is in fact a mixture of declarative and imperative language constructs. Thus, it is difficult to debug a weaving model in terms of weaving operators, because they do not explicitly remain in the model transformation code. Moreover, although it is possible to add new weaving operators, the specification of the operational semantics thereof is cumbersome, since the whole higher-order transformation must be adapted. Finally, a weaving operator always connects source metamodel elements to target metamodel elements, so it is not possible to realize complex transformation logic by the composition of operators.

Abstract Reuse. Abstract reuse in the form of transformation patterns is still in its infancy. A first list of patterns in the context of graph transformations has been proposed by Agrawal et al. [8]. Another initial list of patterns originating from QVT Relations specifications has been collected by Iacob et al. [9]. These two lists can act as an initial input for our pattern language.

Concrete Reuse. Typically, model transformation languages, e.g., ATL [13] and QVT [3], allow to define transformation rules based on types of the corresponding metamodels. Consequently, model transformations are not reusable and must be defined from scratch again and again. One exception is the approach of Varró et al. [14] who define a notion of generic transformations within their VIATRA2 framework, which in fact resembles the concept of templates in C++ or generics in Java. Another approach which is now integrating the idea of genericity are TGGs [15]. Therefore, VIATRA2 as well as TGGs also provide a way to implement reusable model transformations and could be principally used to implement our mapping operators. Nevertheless, they do not foster an easy to debug execution model as is the case with our proposed Transformation Nets.

5 Evaluation

The evaluation of our approach is based on the following four research questions:

Question 1: *Are the found patterns useful/applicable in diverse scenarios?* Concerning this question, the following strategy will be applied. The case studies consisting of numerous transformation examples as described in Section 3 will be divided into a training set and a test set. The training set will be taken for finding recurring transformation patterns. Afterwards the test set will be realized with the found patterns in the training set and evaluated on the basis of corresponding reuse metrics [16].

Question 2: *Does the approach lead to a better understanding of large scenarios?* Regarding this issue, an empirical study will be conducted with students from our model engineering courses (around 200 master students every year). The aim of this empirical study is to evaluate whether the abstract mapping view leads to a better understanding of a large problem. Therefore, the students will be divided into two subgroups, whereby one subgroup gets the transformation definition in our proposed formalism and the other subgroup gets the transformation definition in a low-level transformation language. The understandability will then be evaluated based on questionnaires.

Question 3: *Is productivity of the development process increased by the usage of reusable components?* Concerning this point, again an empirical study will be conducted. Thereby, three distinct transformation approaches will be presented, including our proposed approach. Afterwards the students will have to solve a certain problem with each of these approaches. The productivity will then be evaluated based on corresponding metrics.

Question 4: *Is the quality in the sense of correctness of the resulting model transformations increased by the usage of the reusable components?* Regarding the correctness of the resulting model transformations, also an empirical study will be conducted in conjunction with the study evaluating question 3. Thereby also the quality in terms of freedom from errors will be measured.

6 Current Status

This research effort is still in an initial stage comprising one publication [17] which describes a first set of mapping operators. Furthermore, a complementing research effort realizing the transformation view is currently conducted by [18].

References

1. Bézivin, J.: On the unification power of models. Journal on Software and Systems Modeling **4(2)** (2005) 171–188
2. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal **45**(3) (2006) 621–645
3. Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/-Transformation Specification. www.omg.org/docs/ptc/07-07-07.pdf (2007)
4. Kashyap, V., Sheth, A.: Semantic and schematic similarities between database objects: A context-based approach. The VLDB Journal **5**(4) (1996) 276–304
5. Jensen, K., Kristensen, L.M.: Coloured Petri Nets - Modeling and Validation of Concurrent Systems. Springer (2009)
6. Reiter, T., Wimmer, M., Kargl, H.: Towards a runtime model based on colored Petri-nets for the execution of model transformations. In: 3rd Workshop on Models and Aspects-Handling Crosscutting Concerns in MDSD, Berlin, Germany. (2007)
7. Wimmer, M., Kusel, A., Schoenboeck, J., Kappel, G., Retschitzegger, W., Schwinger, W.: Reviving QVT Relations: Model-based Debugging using Colored Petri Nets. In: Proc. of MoDELS '09, Denver (2009)
8. Agrawal, A., Vizhanyo, A., Kalmar, Z., Shi, F., Narayanan, A., Karsai, G.: Reusable idioms and patterns in graph transformation languages. Electronic Notes in Theoretical Computer Science **127**(1) (March 2004) 181–192
9. Iacob, M.E., Steen, M.W.A., Heerink, L.: Reusable model transformation patterns. Volume 0., Los Alamitos, CA, USA, IEEE Computer Society (2008) 1–10
10. Legler, F., Naumann, F.: A Classification of Schema Mappings and Analysis of Mapping Tools. DB-Systeme in Business, Technologie und Web **12** (2007) 449–463
11. Alexe, B., Tan, W., Velegrakis, Y.: STBenchmark: Towards a Benchmark for Mapping Systems. Proc. of the VLDB Endowment archive **1**(1) (2008) 230–244
12. Fabro, M.D.D., Bézivin, J., Jouault, F., Breton, E., Gueltas, G.: AMW: A generic model weaver. In: Proceedings of the 1ères Journées sur l'Ingénierie Dirigée par les Modèles, Paris, France. (2005) 10
13. Jouault, F., Kurtev, I.: Transforming Models with ATL. Model Transformations in Practice Workshop of MODELS'05 (2005)
14. Varró, D., Pataricza, A.: Generic and meta-transformations for model transformation engineering. In Baar, T., Strohmeier, A., Moreira, A., Mellor, S., eds.: Proc. UML 2004: 7th International Conference on the Unified Modeling Language. Volume 3273 of LNCS., Lisbon, Portugal, Springer (October 10–15 2004) 290–304
15. Amelunxen, C., Legros, E., Schurr, A.: Generic and reflective graph transformations for the checking and enforcement of modeling guidelines. In: IEEE Symposium on Visual Languages and Human-Centric Computing. (2008) 211–218
16. Frakes, W., Terry, C.: Software reuse: metrics and models. ACM Comput. Surv. **28**(2) (1996) 415–435
17. Kappel, G., Kargl, H., Reiter, T., Retschitzegger, W., Schwinger, W., Strommer, M., Wimmer, M.: A framework for building mapping operators resolving structural heterogeneities. In: Proc. of Information Systems and e-Business Technologies (UNISCON'2008), Springer, LNBIP 5 (2008) 158–174
18. Schoenboeck, J.: Transformation Nets - A Runtime Model for Transformation Languages. Doct. Symp., Models 2009, Denver, USA (2009)