

Breaking the Domination of the Internal Graph Model

Florian Heidenreich
Technische Universität Dresden
Software Technology Group
01062 Dresden, Germany
florian.heidenreich@tu-dresden.de

Ulf Wemmie
Technische Universität Dresden
Software Technology Group
01062 Dresden, Germany
s2918002@mail.inf.tu-dresden.de

ABSTRACT

Graph rewrite systems are often build to only transform graphs that are expressed using their internal graph modelling language. This prevents the use of the advanced techniques in graph rewriting on graphs or models that are not expressed in a way that the tool is able to understand. In this paper we present our approach to model migration for graph rewrite systems, that is, adaptation of graphs from external tools to the graph rewrite system's internal model. We exemplify our ideas by a prototypical implementation for Fujaba4Eclipse.

Categories and Subject Descriptors

D.2.2 [Design Tools and Techniques]: Computer-aided software engineering (CASE); D.2.2 [Design Tools and Techniques]: Object-oriented design methods

General Terms

Design, Languages

Keywords

Graph Rewriting, Model Migration

1. INTRODUCTION

One of the major drawbacks of current Graph Rewrite Systems (GRS) is the tight coupling of the system to a specific repository and a specific internal graph model. This situation often prevents use of the GRS in other context that was not foreseen by the developers of the specific GRS. To alleviate this situation, a general approach for adaptation of external models to an internal model of the GRS is crucial. Doing this manually is an error-prone and tedious task. Therefore we aim at providing a semi-automated generative solution for bridging from external models to internal models of the GRS (and back again). We use Fujaba4Eclipse [3] as GRS to exemplify our approach, since it also suffers from the limitations mentioned above.

The paper is structured as follows. At first we elaborate the domain of model migration and present the different steps that are needed to migrate an external model to the internal model of Fujaba. Section 3 highlights interesting points of the prototypical implementation while Section 4 presents related work. Section 5 summarizes the paper and discusses future work.

2. MODEL MIGRATION

As motivated in the introduction, for the exchange of models between different applications an adaptation between source and target is needed. During the adaptation process an *external model*

is transformed to an *internal model*, i.e. the graph model of Fujaba4Eclipse. To access the model of the external application, access to its representation, i.e. in a specific repository, is needed. We call this part of the process *physical adaptation*. In many cases the external application and the GRS (Fujaba in our case) do not share the same metamodel. Therefore a second step is needed, where model elements of the external model are mapped to the target domain. This process of *domain adaptation* is based on the meta-model level.

Our approach for model migration within Fujaba4Eclipse is based on the notion of *adaptation chaining* where a separation of physical and domain adaptation allows for reuse of adapters. Furthermore, the development of the adapters can be done by different software developers where each of them has specific skills regarding repository or domain knowledge. The physical adaptation A_P is done programmatically on source code level, whereas the domain adaptation A_D is defined by Triple Graph Grammars [5, 7] with a graphical notation. The overall adaptation chain is depicted in Figure 1.

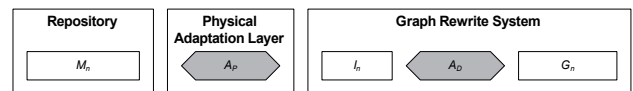


Figure 1: Adaptation chain of physical and domain adaptation.

2.1 Physical Adaptation

A physical adaptation A_P is an exogenous transformation [6] that transforms the source model¹ M_n of the external tool into an intermediate model I_n , where I_n acts as an interface between physical and domain adaptation. I_n still represents the concepts of the external model but is modelled using concepts of the graph rewrite system. To transform the model elements from the repository to the representation of I_n the physical adaptation layer from Figure 1 is refined into an adaptation chain consisting of *repository adaptation* A_{Rep} and *element adaptation* A_{Elem} (see Figure 2).

The bidirectional repository adapter provides a linking to the repository of the external application. It offers means to both access existing models and to create new models in the repository. The intermediate model I_n conforms to the metamodel M_{n+1} of the external model M_n but uses Fujaba classes and interfaces for its implementation of the element adapters. Thereby, persistence and the uniform processing of the intermediate model is assured. Since

¹In the following we use the abbreviations M for the external model, I for the intermediate model, and G for the internal graph model.

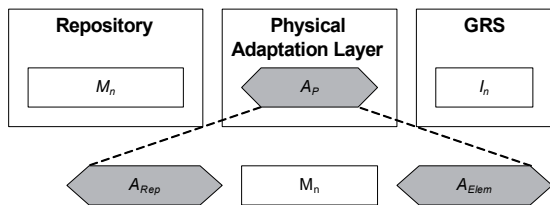


Figure 2: Physical adaptation refined to adaptation chain of repository and intermediate adaptation.

the number of metalevels is not fixed in general, one can imagine migration of models on multiple metalevels. Since Fujaba offers levels for metamodels, models and objects (besides its internal meta-metamodel) the specification of graph-rewrite rules is limited to the model level. This observation leads to a total of two metalevels that can be migrated within Fujaba, where the first level is subject for transformation through the GRS.

2.2 Domain Adaptation

Domain adaptation is an endogenous transformation [6] that provides a bidirectional mapping of elements from the metamodel I_{n+1} of the intermediate model I_n to the Fujaba metamodel. The general idea is to map elements from one domain to another domain, where the specific elements are playing *identical roles* in the specific domains. To exemplify this, the element `Eclass` from the Eclipse Metamodeling Language Ecore [2] can be mapped to Fujaba's `UMLClass`, since both elements are playing the roles of the class concept in their domain.

2.3 Process Overview

While the last sections show the general ideas behind our approach to model migration, this section presents an overview to the overall process from the adaptation of the external application to the successful transformation of the adapted model in Fujaba4Eclipse.

First, a Fujaba conforming representation of the meta-metamodel² of the external application is needed. This representation can be modelled as Fujaba class diagram and afterwards generated to Java code. This is necessary because initially there are no means to import models of the external application to Fujaba. This initial step is depicted in Figure 3.

The modelled meta-metamodel acts as an interface between physical and domain adaptation. It is used to generate the physical representation of the external meta-metamodel as well as to the partial

²The indices refer to the standard metamodel levels of the MOF metalevel architecture.

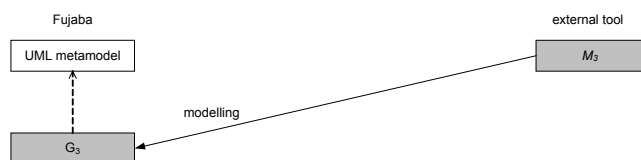


Figure 3: Modelling the meta-metamodel of the external application.

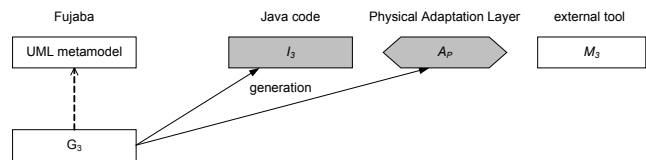


Figure 4: Generation of the code representation of the meta-metamodel and of the physical adapter.

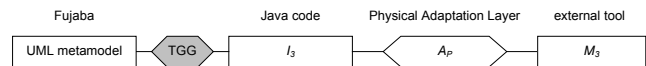


Figure 5: Specification of the domain adaptation.

generation of the physical adapter. Figure 4 shows the generation of the meta-metamodel to Java code and the generation of the physical adapter.

After that, domain adaptation can be specified. Therefore the UML metamodel of Fujaba and the generated physical representation of the external meta-metamodel are used. Triple Graph Grammars [5, 7] are utilised to graphically define the relationship between the different domains. These bidirectional rules are then transformed to Story Diagrams (Fujaba's notion of graph-rewrite rules) that are used to exchange models between the different metamodels (see Figure 5).

This complete adaptation chain now provides means for import and export of metamodels of the external application (see Figure 6). This occurs one metalevel below the metalevel on which the definition of the adapters was done. On import, the metamodel of the external application is transformed to the intermediate representation using the physical adapter. The intermediate representation is then transformed into a Fujaba UML class diagram using the rules that were specified using Triple Graph Grammars.

It is obvious, that an additional adapter is needed to process instances of the imported metamodel in Fujaba. But in contrast to the development of the previous adapter, the metamodel already exists as a UML class diagram and does not need to be modelled anymore. The next steps that are needed to generate an adaptation chain are straight-forward according to the previous procedure (see Figure 7). In the last step, a physical adapter to import instances of M_2 needs to be developed. This adapter allows for applying the GRS's rewriting capabilities on the imported models.

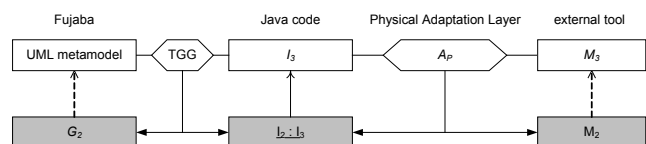


Figure 6: Migration of metamodels.

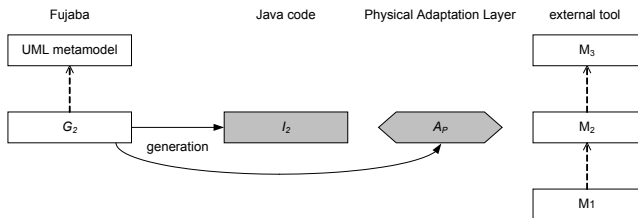


Figure 7: Initial situation for further adaptation.

3. IMPLEMENTATION

We have build a prototype implementation of the presented approach using *Fujaba4Eclipse* and the plug-ins *TGGEdition* (editor for Triple Graph Grammars) and *MoTE* (Model Transformation Engine). It uses the extension points offered by the three components. The prototype itself provides additional extension points, namely *PhysicalAdapter* and *DomainAdapter* for the adaptation of external models and *MigrationAdapter* for the configuration of the interaction of the physical and the domain adapter.

However, the generation of the adaptation chain is still in a very early stage and we try to investigate in further improvements with regards to the automation of the process.

4. RELATED WORK

The *Tool Adapter* proposed by Kindler and Wagner [5] is strongly related to our approach. The authors present an adaptation architecture for adaptation of proprietary metamodel implementations of external tools to the metamodel implementation of the standalone version of the Fujaba tool suite. As we do, they also use TGGs for domain adaptation but use hand-written adapters to adapt the external metamodel (while we follow a semi-generative approach). In contrast to our work, they do not use a dedicated intermediate metamodel on which the adapter is built upon, but transform the Fujaba-generated metamodel of the external application in an adapter to the external metamodel by replacing the bodies of accessor methods with delegation code to the API of the external application. This, on the one side, improves performance of the adaptation, but, on the other side, decreases flexibility in exchanging and reusing existing adapters.

Another approach that is related to our work is the concept of non-materialized model view specifications based on a extension of TGGs—the declarative view specification approach VTTG [4]. Instead of copying tool data and creating physical representations of the target model, the authors present an approach that offers virtual views of models that can be manipulated and are synchronised automatically.

The *Atlas Model Weaver* [10] offers a means for domain adaptation, where the adaptation is specified by a dedicated *weaving model*. This weaving model is used to generate model transformations from one domain to another domain. The approach is very similar to domain adaptation by Triple Graph Grammars but lacks its mathematical foundations [7].

The *Tiger EMF Transformation Project* [1, 11] is a framework for EMF transformations based on graph transformation. It supports the definition of graph-rewrite rules on arbitrary EMF-based meta-

models and uses AGG [8, 9] as GRS. Internally, the imported EMF models are transformed to an AGG representation in a fully automated way. This high degree of automation is at the cost of flexibility regarding the supported external applications (which is Eclipse EMF in this case).

5. SUMMARY

In this paper we presented our work on model migration for graph rewrite systems, that is, mapping external models to internal models of the GRS and back again and presented a general architecture for this. Our approach is based on adaptation chaining, where multiple adapters are chained together and where each of them takes a specific role in the process of model migration. The decomposition of the adaptation problem allows for partial generation of the adapters—and more importantly—for reuse of adapters. We built a prototypical implementation based on *Fujaba4Eclipse*, which enables usage of Fujaba’s advanced graph rewriting techniques on models from other applications in a systematic way.

In our future work we want to improve the generation of the adaptation chains through annotations in the modelled metamodels that would allow for an fully-automated mapping of the external model elements to the GRS’s internal representation.

6. ACKNOWLEDGEMENTS

This research has been co-funded by the German Ministry of Education and Research (BMBF) within the project *feasiPLe*³.

7. REFERENCES

- [1] E. Biermann, K. Ehrig, C. Köhler, G. Kuhns, G. Taentzer, and E. Weiss. Graphical definition of in-place transformations in the eclipse modeling framework. In O. Nierstrasz, J. Whittle, D. Harel, and G. Reggio, editors, *MoDELS*, volume 4199 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 2006.
- [2] F. Budinsky, S. A. Brodsky, and E. Merks. *Eclipse Modeling Framework*. Pearson Education, 2003.
- [3] Fujaba Project Team. *Fujaba4Eclipse*, Aug. 2007. URL <http://www.fujaba.de>.
- [4] J. Jakob, A. Königs, and A. Schürr. Non-materialized model view specification with triple graph grammars. In *Proceedings of the 3rd International Conference on Graph Transformations (ICGT'06)*, volume 4178 of *LNCS*, pages 321–335. Springer, 2006.
- [5] E. Kindler and R. Wagner. *Triple Graph Grammars: Concepts, Extensions, Implementations, and Application Scenarios*. Technical Report tr-ri-07-284, Software Engineering Group, Department of Computer Science, University of Paderborn, June 2007.
- [6] T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, Mar. 2006.
- [7] A. Schürr. Specification of graph translators with triple graph grammars. In *Workshop on Graph-Theoretic Concepts in Computer Science*, pages 151–163, 1994.
- [8] G. Taentzer. AGG: A Graph Transformation Environment for System Modeling and Validation. In *Tool Exhibition at Formal Methods 2003*, 2003.
- [9] The AGG Project Team. *AGG: The Attributed Graph Grammar System*, Aug. 2007. URL <http://tfs.cs.tu-berlin.de/agg/>.
- [10] The AMW Project Team. *Atlas Model Weaver*, Aug. 2007. URL <http://eclipse.org/gmt/amw/>.
- [11] Tiger EMF Transformation Project Team. *Tiger EMF Transformation Project*, Aug. 2007. URL <http://tfs.cs.tu-berlin.de/emftrans/>.

³cf. <http://www.feasiple.de>