Applying Bidirectional Transformations in Industrial Contexts: Challenges and Solutions

Nils Weidmann, Stefan Sauer
Paderborn University, Department of Computer Science
Fürstenallee 11, 33102 Paderborn, Germany
{nils.weidmann,sauer}@upb.de

Abstract

Bidirectional Transformation (bx) has been a well-known approach for maintaining consistency between software models for many years. Due to their sound formal basis, model transformations have gained broad acceptance in academia. However, the applicability of model transformation approaches in industrial contexts is often not that obvious at the first glance. Due to an increasing, sophisticated tool support for the formal concepts, use cases emerged that give evidence for the beneficial use of bx in industry. This paper briefly presents examples for such use cases, the bx tools in use and how further research can address open challenges in the field of Model-Driven Engineering (MDE).

Keywords

bidirectional transformation, triple graph grammar

1 Introduction and Motivation

In the field of MDE, the evolution of models and the resulting challenge of maintaining consistency between semantically interrelated models plays an important role. In this regard, bx allows to perform multiple consistency management operations such as forward and backward transformation, synchronization and consistency checking based on the same consistency relation. In recent years, an interdisciplinary community has grown - involving researchers from the fields of e.g. functional programming, databases and MDE - which proposes concepts and tools for bx. Although the formal concepts are sound and tools are continuously maintained, the question arises if bx frameworks are directly applicable in industrial contexts, facing hard requirements regarding efficiency, maintainability, usability and user acceptance. We underpin the usefulness of bx approaches by briefly presenting selected examples where Triple Graph Grammars (TGGs) - as a representative for bx approaches from the MDE area - are successfully applied to real-world industrial use cases (Sect. 2). After a brief introduction to TGGs (Sect. 3), eMoflon[1] and MoTE[2] as representatives for bx tools under active development are sketched (Sect. 4), before open research challenges and possible solutions are discussed to conclude this paper (Sect. 5).

2 Industrial Use Cases

The first case study originates from a project conducted by Siemens AG and TU Darmstadt [5]. In the engineering domain, the development of physical components used in e.g. excavators or water supply lines is distributed among different teams of engineers, who work concurrently on models describing these components from different view points. The use case involves the synchronisation of a Computer-Aided Design (CAD) model maintained by mechanical engineers and a simulation model maintained by electrical and automation engineers. Both models have a relatively small semantic overlap, which means that both teams can work largely independent from each other. For maintaining consistency, a sequence of consistency checks and change propagations is necessary, whereby the consistency relation is bidirectionally expressed via TGG rules.

Secondly, a bx approach was recently implemented to solve an allocation problem between software testers and testing tasks during testing phases for software releases for automotive development tools at dSPACE GmbH in cooperation with Paderborn University. Prior to the project, a test manager has planned the assignment of testers to test tasks manually, which was replaced by a software solution relating a testing team model to a task model. The special challenge in this case was the consideration of different constraints, such as the availability of employees per time period, the suitability of employees for different tasks based on their qualification, and dependencies between tasks induced by the testing process. Additionally, it is desirable to maximize the available buffer times, especially for critical tasks. For defining constraints and objectives, a Domain-Specific Language (DSL) was developed which should be manageable by the test manager without knowledge about the underlying technology. The approach enables the test manager to automatically create an initial proposal for the test plan, whereby manual adaption and the consistent propagation of those are possible afterwards.

3 Maintaining Consistency by means of TGGs

TGGs are a declarative and rule-based means of maintaining consistency between two semantically interrelated models. Besides these models, which are (interchangeably) denoted as *source* and *target*, a third *correspondence* model is introduced that sets up the consistency relation. Models are represented as attributed graphs, while nodes and edges are typed to establish the conformance to the respective metamodels. To make this more concrete, we recall the second case study from Sect. 2. A rule that relates a software tester to the execution of a testing task is depicted in Fig. 1. In the source model (depicted on the left-hand side), a person which is available in a particular week, as well as an execution environment (i.e., a computer with pre-installed software) must already exist to apply the rule. In the target model, only the testing task to be performed is required. The rule creates a new execution node for this task in the target model and links it to the availability and the environment via correspondence nodes, which are depicted as hexagons here. The created nodes and edges are green and annotated with a ++ mark-up, whereas the required elements, which we denote as *context* elements, are coloured black. Besides the structural changes, an attribute condition in textual form ensures that the duration of the task does not exceed the availability of the tester.

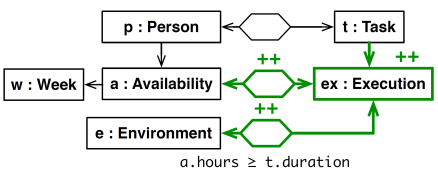

Figure 1: TGG Rule `AddFirstExecution`

[1] emoflon.org [2] bit.ly/2Xct0tF

These declarative rules can be operationalised for various tasks. Instead of creating all green elements anew, elements of given models can be marked as translated instead, depending on the concrete consistency management task. For forward (backward) transformation, the source (target) model is already given, such that one searches for a rule application sequence which builds up the given model, simultaneously creating the transformed model of the respective other domain. Similarly, the goal of consistency checking is to find a triple that relates the given source and target models and can be built up by sequenced rule applications of the TGG.

4 Tool Support

As a cross-cutting community involving researchers from various disciplines actively works on bx topics, a substantial amount of tooling[3] has been proposed which cannot be presented in this place. Instead, we focus on a brief introduction to two exemplary TGG tools under active development.

eMoflon::IBeX [9], which is used in both case studies described in Sect. 2, is a plug-in building up on the Eclipse Modeling Framework (EMF) that compiles the specified TGG rules into Java code at design time. Multiple external frameworks such as Xtext[4] for code generation and PlantUML[5] for read-only visualization of rules and (meta)models are used. Along with adapters for incremental pattern matchers and Integer Linear Programming (ILP) solvers, a modular architecture is set up that allows to exchange components if needed. As an alternative to storing models as Ecore-compliant XML Metadata Interchange (XMI) files, a solution which connects the graph database Neo4J[6] is developed in order to store models, metamodels and their conformance relation in a uniform manner.

Another TGG tool with a strong focus on efficiency is MoTE, which is also implemented in form of an Eclipse plug-in. With this tool, it was possible to build a realistic synchroniser for Analysis and Design Language (AADL) and Simulink models [1], which are standards used in the embedded systems domain. As an extension to the original TGG formalism, MoTE supports the connection of more than one element per model with a correspondence node, which turned out to be very beneficial for the case study to keep the number of TGG rules small. Compared to eMoflon, the support for advanced language features is quite limited, though, restricting the expressiveness of supported TGGs.

5 Open Challenges and Future Work

Whereas basic operations such as forward and backward transformation, synchronization and consistency checking are well-supported already, there are also open issues whose solution would broaden the practical usage of bx approaches.

Techniques for model synchronizations are currently applicable in cases where only one model is changed, such that the delta can be propagated to the respective other domain. An interesting and simultaneously relevant scenario would be the merge of such deltas resulting from concurrent modifications. A three-way-merge algorithm known from different versioning systems does not address this problem sufficiently as more than one metamodel is involved and the outcome depends on which of the two deltas is propagated first [6]. More promising seems to be a detection of conflicting changes and a classification of conflicts that can occur. Based on this, policies can be introduced that specify how a conflict is resolved dependent on its type. Although seminal work has already been proposed by Fritsche et al. [3], it is still challenging to define the desirable outcome of such a model integration process, even for smaller examples, contrasting the straightforward solution for transformation and synchronization tasks.

When shifting the focus from ideal settings to realistic use cases, further research on fault-tolerant bx gets more and more important. Assuming and enforcing strictly consistent models and changes at each point of time is unrealistic for several reasons [7], and not even compliant with what a user expects from a consistency management system. Therefore, the binary distinction between success and failure of a transformation (or between a consistent and inconsistent result of a check) should be turned into an optimisation problem which can take consistency, but also further objectives into account, such as surprising the user with automated changes only as much as absolutely necessary [2]. Consistency checking as well as forward and backward transformation have already been modelled as search problems [10], but a proper representation of inconsistencies and error handling facilities for the user are still an open issue.

Finally, recent work pointed out that there is indeed a difference between an n-ary consistency relation and the union of respective binary consistency relations between each pair out of these n models [8]. Conceptual approaches to overcome these problems have already been developed [4], but further tool support and large-scale evaluations are needed to properly address the new challenges of multidirectional transformations.

[3] bx-community.wikidot.com/relatedtools
[4] www.eclipse.org/Xtext/
[5] plantuml.com
[6] neo4j.com

References

[1] Dominique Blouin, Alain Plantec, Pierre Dissaux, Frank Singhoff, and Jean-Philippe Diguet. 2014. Synchronization of Models of Rich Languages with Triple Graph Grammars: An Experience Report. In *Theory and Practice of Model Transformations - 7th International Conference, ICMT 2014 (Lecture Notes in Computer Science)*, Davide Di Ruscio and Dániel Varró (Eds.), Vol. 8568. Springer, 106–121.

[2] James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. 2017. On principles of Least Change and Least Surprise for bidirectional transformations. *Journal of Object Technology* 16, 1 (2017), 3:1–31.

[3] Lars Fritsche, Jens Kosiol, Andy Schürr, and Gabriele Taentzer. 2019. Efficient Model Synchronization by Automatically Constructed Repair Processes. In *Fundamental Approaches to Software Engineering - 22nd International Conference, FASE 2019 (Lecture Notes in Computer Science)*, Reiner Hähnle and Wil M. P. van der Aalst (Eds.), Vol. 11424. Springer, 116–133.

[4] Heiko Klare and Joshua Gleitze. 2019. Commonalities for Preserving Consistency of Multiple Models. In *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019*, Loli Burgueño, Alexander Pretschner, Sebastian Voss, Michel Chaudron, Jörg Kienzle, Markus Völter, Sébastien Gérard, Mansooreh Zahedi, Erwan Bousse, Arend Rensink, Fiona Polack, Gregor Engels, and Gerti Kappel (Eds.). IEEE, 371–378.

[5] Erhan Leblebici. 2018. *Inter-Model Consistency Checking and Restoration with Triple Graph Grammars.* Ph.D. Dissertation. Darmstadt University of Technology, Germany.

[6] Fernando Orejas, Artur Boronat, Hartmut Ehrig, Frank Hermann, and Hanna Schölzel. 2013. On Propagation-Based Concurrent Model Synchronization. *ECEASST* 57 (2013).

[7] Perdita Stevens. 2014. Bidirectionally Tolerating Inconsistency: Partial Transformations. In *Fundamental Approaches to Software Engineering - 17th International Conference, FASE 2014 (Lecture Notes in Computer Science)*, Stefania Gnesi and Arend Rensink (Eds.), Vol. 8411. Springer, 32–46.

[8] Perdita Stevens. 2017. Bidirectional Transformations in the Large. In *20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2017*. IEEE Computer Society, 1–11.

[9] Nils Weidmann, Anthony Anjorin, Lars Fritsche, Gergely Varró, Andy Schürr, and Erhan Leblebici. 2019. Incremental Bidirectional Model Transformation with eMoflon: : IBeX. In *Proceedings of the 8th International Workshop on Bidirectional Transformations co-located with the Philadelphia Logic Week, Bx@PLW 2019 (CEUR Workshop Proceedings)*, James Cheney and Hsiang-Shang Ko (Eds.), Vol. 2355. CEUR-WS.org, 45–55.

[10] Nils Weidmann, Anthony Anjorin, Erhan Leblebici, and Andy Schürr. 2019. Consistency management via a combination of triple graph grammars and linear programming. In *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2019*, Oscar Nierstrasz, Jeff Gray, and Bruno C. d. S. Oliveira (Eds.). ACM, 29–41.