Author version of the accepted manuscript. Final publication: M. E. Kramer. "Synchronizing Heterogeneous Models in a View-Centric Engineering Approach". In: *Software Engineering 2014 – Fachtagung des GI-Fachbereichs Softwaretechnik*. Ed. by W. Hasselbring and N. C. Ehmke. Vol. 227. GI Lecture Notes in Informatics. Doctoral Symposium. Kiel, Germany: Gesellschaft für Informatik e.V. (GI), 2014, pp. 233–236

Synchronizing Heterogeneous Models in a View-Centric Engineering Approach

Max E. Kramer
Karlsruhe Institute of Technology
max.e.kramer@kit.edu

Abstract:
When software systems are modelled from different viewpoints using different notations, it is necessary to synchronize these heterogeneous models in order to sustain consistency. To realize this for a specific system, developers need two competences: They have to express the conceptual relationships between the elements of different modelling languages and domains. But they also have to master various techniques of Model-Driven Engineering (MDE), such as transformation languages. Current synchronization approaches, however, do not address these requirements separately and mix technical and conceptual challenges. To ease heterogeneous modelling, we propose a view-centric engineering approach, in which incremental synchronization transformations are generated from abstract synchronization specifications. This will make it possible to declare mappings and invariants for model concepts with a domain-specific language, for which developers can reuse and customize technical solutions of a generator. In this paper, we introduce the according research goals and questions and sketch our plans for realization and evaluation.

1 Introduction and Motivation

Many software systems are modelled from different viewpoints using different modelling languages in order describe and analyse the systems appropriately for specific tasks. On the one hand, these different models may conform to different metamodels and on the other hand, they may describe identical parts of a system. A component-based system, for example, may be modelled in an abstract way with an architectural description language, while the structure and behaviour of individual components may be modelled with UML class and sequence diagrams. Because of the semantic overlap between these models, it is necessary to keep them consistent.

Current approaches for the synchronization of heterogeneous models, however, mix the conceptual challenge of identifying and expressing relationships between domain elements with technical challenges of transformation techniques. In this paper, we present our plans for a synchronization language for a view-centric engineering approach that separates technical solutions from conceptual synchronization specifications. It uses declarative mappings between metaclasses, normative invariants and imperative transformation customization code. From these three language parts, incremental synchronization transformations are derived using a customizable generator. Moreover, synchronization specifications may be used for the definition of views that integrate information from multiple metamodels and vice-versa.

2 Background and Related Work

In Model-Driven Engineering, models have to conform to a metamodel in order to be processed in transformations. This syntactic constraint makes it possible to derive source code, documentation, and other artefacts from appropriate models.

To cope with different views and notations, various model synchronization approaches have been presented. The multi-view point approach [RJV09], for example, defines direct correspondences between views. For such synthetic approaches, the link complexity grows exponentially with the number of views. In contrast to this, projective approaches like Orthographic Software Modelling [ASB10] synchronize views with a central model. Such a central model limits the expressive power and has to be designed upfront with all possible views, which makes evolution and extensions difficult and hinders support for legacy metamodels and views. Tool integration approaches like ModelBus [HRW09] provide advanced features like model merging but lack concepts for expressing the semantic relations between metamodels. Triple-Graph Grammars (TGGs) have been successfully used for model synchronization, for example, for SysML and AUTOSAR [GHN10]. The morphisms that map an interface part of a TGG to the left and right hand side are similar to our mappings. But TGGs do not separate technical transformation details from domain concepts.

3 Goals and Questions

Our plans for model synchronization are a part of the view-centric Vitruvius approach [KBL13]. They are led by the following two research goals: *G1: Ease the synchronization of heterogeneous models in a way that sustains consistency between those models. G2: Support the definition of integrated views on heterogeneous models through the use of synchronization information and vice-versa.*

We pursue these goals by answering the following two research questions: *Q1: Can we synchronize heterogeneous models automatically using transformations that are generated from abstract synchronization specifications? Q2: Can we couple view type definitions and model synchronization specifications?* Each of these questions can be divided into three subquestions: *Q1.1: Can we specify synchronization with a precise and expressive domain-specific language based on declarative mappings, normative invariants and custom response transformation snippets? Q1.2: Can we generate synchronization transformations from these mappings and embed the response snippets? Q1.3: Can we synchronize models by triggering these transformations after atomic changes and after invariant violations? Q2.1: Can we derive synchronization specifications from view type specifications and vice-versa? Q2.2: Can we restrict editability in view type specifications according to synchronization specifications? Q2.3: Can we derive synchronization requirements from view type specifications?*

4 Approach

We will answer our research questions by developing and evaluating a Domain-Specific Language (DSL) for synchronization specifications within the Vitruvius approach [KBL13]. A generator for this DSL will produce incremental transforma-

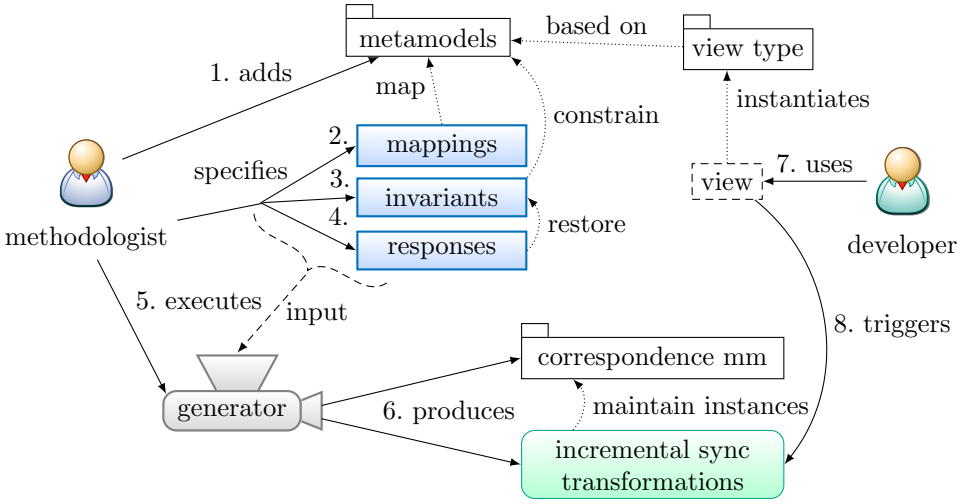

Figure 1: Process for defining and executing synchronizations in view-centric engineering

tions from specifications. Before we explain the language and its use, we sketch the overall approach and mention an important assumption: The approach combines all information of a software system in a virtual single underlying model that can be accessed solely by views conforming to view types. All views have to report sequences of atomic changes if changes that shall be persisted and propagated occurred in a view. The incremental transformations that we generate from the DSL are triggered using these atomic changes. In general, they cannot be derived unambiguously from model differences but have to be recorded using customized listeners, which may be based on refactoring commands for textual languages. The virtual model is a modular combination of individual models conforming to different metamodels. This decoupled layout makes it possible to integrate legacy metamodels and allows for independent evolution. The definition of view types and the integration of the modular metamodel are carried out by a special role called *methodologist*.

The synchronization language consists of three parts for *specifying*, *checking* and *preserving* consistency. In the first part, declarative *mappings* specify semantic correspondences between metaclasses, their attributes, and references using conditions that are based on attribute and reference values. In the second part, consistency checks within and between models may be defined with normative *invariants* that are formulated using the Object Constraint Language (OCL). Invariants may be specified for individual metamodels or combinations thereof and may expose parameters that can be used by the third language part. In this part *responses* to specific modifications or invariant violations can be encoded using a general-purpose model transformation language (QVT-O or ATL). Thus the power and expressivity of a general-purpose language can be used if the language constructs for synchronization mappings are not sufficient, for example, for clean-up actions or conflict resolution.

The detailed process for defining and executing synchronization behaviour using the language and generator is shown in Figure 1. The methodologist, who is responsible for the virtual metamodel, view types, and synchronization, first adds metamodels to the virtual metamodel. Then, he specifies mappings between metamodels, invariants that constrain these metamodels, and optional responses that may restore these invariants. Afterwards, the methodologist executes a generator, which produces a correspondence metamodel and incremental sync transformations from the three specification parts. For every mapped metaclass and every modification type a separate transformation, which embeds response transformation snippets, is produced. These transformations are triggered if a developer changes instances of the corresponding metaclasses in a view conforming to a view-type.

5 Evaluation

We will evaluate our synchronization language and generator in a case study that combines a component-based Architectural Description Language (ADL), UML class diagrams and Java source code. With this case study, we evaluate whether it is possible to synchronize architectural models, class diagrams and source code with our approach. In order to assess the benefits of our approach with respect to *G1*, we are planning to conduct a quasi-experiment with graduate students and engineers. In it, subjects will design and implement component-based software and (a) keep all artefacts manually in sync, (b) use a general-purpose transformation language or (c) our synchronization language and generator. Consistency and effectivity will be measured, but because of the duration we cannot control all influencing variables.

References

[ASB10] C. Atkinson, D. Stoll, and P. Bostan. "Orthographic Software Modeling: A Practical Approach to View-Based Development". In: *Evaluation of Novel Approaches to Software Engineering*. Vol. 69. Communications in Computer and Information Science. Berlin/Heidelberg: Springer, 2010, pp. 206–219.

[GHN10] H. Giese, S. Hildebrandt, and S. Neumann. "Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent". In: *Graph Transformations and Model-Driven Engineering*. Vol. 5765. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, pp. 555–579.

[HRW09] C. Hein, T. Ritter, and M. Wagner. "Model-Driven Tool Integration with ModelBus". In: *Workshop Future Trends of Model-Driven Development*. 2009.

[KBL13] M. E. Kramer, E. Burger, and M. Langhammer. "View-centric engineering with synchronized heterogeneous models". In: *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. VAO '13. Montpellier, France: ACM, 2013, 5:1–5:6.

[Kra14] M. E. Kramer. "Synchronizing Heterogeneous Models in a View-Centric Engineering Approach". In: *Software Engineering 2014 – Fachtagung des GI-Fachbereichs Softwaretechnik*. Vol. 227. GI Lecture Notes in Informatics. Doctoral Symposium. Kiel, Germany: Gesellschaft für Informatik e.V. (GI), 2014, pp. 233–236.

[RJV09] J. R. Romero, J. I. Jaén, and A. Vallecillo. “Realizing Correspondences in Multi-viewpoint Specifications”. In: *Enterprise Distributed Object Computing Conference, 2009. EDOC ’09. IEEE International*. 2009, pp. 163–172.