Using Conceptual Modeling for Designing Multi-View Modeling Tools

Full paper

Dominik Bork
Research Group Knowledge Engineering
University of Vienna
Vienna, Austria
dominik.bork@univie.ac.at

Abstract

Multi-view modeling methods (MVMM) can cope with the increasing complexity of today's enterprise and information systems by decomposing the corresponding model into several viewpoints. The combination of the instantiated views gives the whole model of the system. Modeling tools are vital for efficient utilization of MVMMs. However, sufficient support in the conceptual design of multi-view modeling tools is not given. The paper at hand introduces the MuVieMoT modeling method, dedicated towards the conceptualization of multi-view modeling tools. The method is focused on how to capture, with conceptual modeling means, the constituents of MVMMs, in terms of viewpoints, modeling procedure and consistency mechanisms specification. The method is aimed at method engineers and tool developers, bridging the gap between tool design and tool development. Applicability of the method is illustrated by a case study, hereby defining the conceptual design of a multi-view modeling tool for an enterprise modeling method.

Keywords

Multi-view modeling, conceptual modeling, MuVieMoT, modeling tool development, model-driven development

Introduction

When creating models for complex real-world or postulated phenomenon, e.g., enterprise models or enterprise architectures, one is confronted with the requirement of managing the inherent complexity. Recurring structuring approaches decompose the complex representation along multiple dimensions into vertical and/or horizontal layers, viewpoints, levels etc. (cf. MEMO (Frank, 1994), SOM (Ferstl and Sinz, 2013), Zachman (1987)). Multi-view modeling is one particular approach for coping with this complexity by decomposing the models into several viewpoints. The viewpoints thereby serve different purposes, e.g., analysis, specification, and design of complex information systems; or different stakeholders, e.g., managers, process owners; realize different abstraction layers, e.g., functional, technical. Consequently, a viewpoint comprises a specific modeling language, modelers utilize to create instances of a viewpoint, i.e., views.

Efficient application of multi-view modeling methods (MVMM) considerably depends on the availability of corresponding modeling tools. However, the specification of such tools is not regarded in research in depth up to now (Frank et al., 2014, p. 41). Formalized specifications of the modeling language of the viewpoints (e.g, using meta models) and the dependencies between according views (e.g., using constraint languages) are not given by the majority of MVMMs (Sinz, 1996; Akehurst, 2004) which makes it hard if even possible to design and implement good tools. Conventional software development approaches and meta modeling platforms lack in supporting the specific requirements of multi-view modeling tool design. Moreover, commonly used modeling languages, e.g., the UML family of languages, are not capable to capture the specifics of MVMMs in an adequate abstraction level due to their generality.

The aim of the paper at hand is to bridge that research gap by introducing MuVieMoT, a modeling method, specifically tailored to the requirements of multi-view modeling methods. The method can be utilized - by means of conceptual modeling - to guide method engineers and tool developers during the conceptualization of multi-view modeling tools. The created models serve different purposes: formal description of MVMMs aspects for understanding and communication, specification of Conceptual Designs for the development of multi-view modeling tools, and definition of usage scenarios of the multi-view modeling tools by modelers. Referring to Bork and Fill (2014), a formal description is referred to as one being intersubjective understandable and machine-processable.

The rest of the paper is organized as follows: First, the theoretical foundation for the application of conceptual modeling towards the conceptualization of multi-view modeling tools is introduced. The focus of the following section is on the components of the MuVieMoT method; mapped to the introduced requirements and specific challenges of the domain. A case study from the enterprise modeling domain then illustrates the application of the method, hereby serving as an evaluation of the methods' utility. Finally, the paper is concluded with a discussion of related works and a SWOT analysis.

Problem Statement and Background

Motivation, Context and Research Challenges

Although there is general agreement on the utility of multi-view modeling methods, and on the efficiency of using modeling tools, there is still a research gap between the abstract constituents of a modeling method and the functionality of a modeling tool (Akehurst, 2004; Bork and Sinz, 2013). Conceptual modeling and model-driven development approaches foster specification and development of software systems by providing modeling concepts, derived from the application domain, and by raising the abstraction level (cf. Mylopoulos (2008) and Marincic et al. (2013)). Conventional procedure models like the Waterfall Model or Prototyping lack at the specifics of MVMM and the transformation of these specifics into requirements; and finally into technical implementation. Moreover, general-purpose modeling languages like the UML lack in specificity in the MVMM application domain.

The solution proposed in this paper builds on a generic modeling method framework. According to Karagiannis and Kühn (2002), a modeling method is composed of three parts: a Modeling Language, a Modeling Procedure, and Mechanisms & Algorithms. A **Modeling Language** defines the elements of the modeling method and the rules for combining them by means of syntax, semantics, and notation (commonly formally specified using meta models (Bork and Fill, 2014)). The **Modeling Procedure** then uses the specified modeling language and defines steps a modeler performs in order to create valid models. The combination of modeling language with modeling procedure realizes the **Modeling Technique** of a modeling method. **Mechanisms & Algorithms** *"provide the functionality to use and evaluate the models built by using the modeling language"* (Karagiannis and Kühn, 2002, p. 185).

Multi-View Modeling

By reverting to multi-view modeling methods, multiple structural and behavioral aspects can be represented using different, inter-connected viewpoints. A viewpoint specifies the modeling language of its views; composed of semantics, syntax and notation. The relationship between viewpoint and view is considered analogous to the one between meta model and corresponding model in the following, i.e., a view is considered an instance of a viewpoint. Over the last decades, multi-view modeling approaches have been successfully applied in different domains (see Kheir et al., (2013) for a comprehensive overview). All application domains utilize the fundamental advantage of MVMM: coping with the complexity of a system by providing specialized viewpoints that decompose the overarching representation, concentrate only on certain aspects while omitting others, and serve specific needs. Consequently, the complexity of creating a view is reduced significantly. The combination of the views then gives the whole model of the system.

Due to overlapping areas of the viewpoints (cf. Nuseibeh et al. (1994) for an overview), consistency between the views is vital for the efficient application of MVMMs. As a consequence, dependencies and synchronization mechanisms need to be specified, ensuring a consistent model state. Figure 1 illustrates

multi-view modeling in a dynamic manner. The specification uses a comprised form of the FDMM formalism (Fill et al., 2012).

Given the viewpoints (**VP_j**, **VP_k**), an instantiation function (**μ(Viewpoint, state)**) instantiates a viewpoint into a view, describing a concrete state of the modeled system. Hence, codifying the modeling constructs and their attribute values based on the viewpoint specification and the knowledge base, describing the state to be modelled. The created views (**v1**, **v2**) are realized as graphical models, representing an instance of a viewpoint. The operation **Op()** alters the content of a view (**v**), thus shifting it in another state (**v`**).

The challenge of multi-view modeling tools is to realize consistency between the views (indicated by the dotted lines between v_1 and v_2, and v_1‘ and v_2’). This consistency can generally be ensured in two ways, either by translating the whole state of the view into another view (referred to as **state translation**, **T_S**) or translating each modeling operation (Op) applied to a certain view into semantically preserving operations that need to be applied to other views (referred to as **transition translation**, **T_T**).

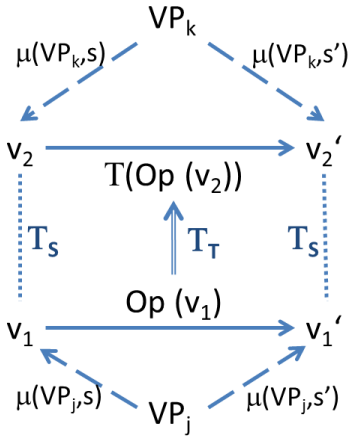

Figure 1. Consistency in Multi-View Modeling

Research Challenges

The contribution of this paper is the MUVIEMOT modeling method. The method enables conceptual modeling of multi-view modeling tools based on corresponding multi-view modeling methods. The following research challenges constitute the building blocks of the method:

- RC-1: What are the specific characteristics of multi-view modeling methods and how can they be transformed into requirements for a modeling method, aiming to codify them in a formalized way?
- RC-2: How to foster, by means of conceptual modeling and meta modeling, the conceptualization of multi-view modeling tools?

Conceptual Modeling of Multi-View Modeling Tools: The MUVIEMOT Method

Research Approach

The MUVIEMOT method has been realized following the design science research (DSR) paradigm (Hevner, 2004). More specifically, the method proposed by Peffers et al. (2007, p. 54f) is used as a methodical framework, thereby structuring the tasks to be performed. Peffers et al. (2007) described a sequence of six tasks. For each task, its corresponding coverage in the paper is as follows: identify problem & motivate (Section 1 and Section 2), define objectives of a solution (Sections 2.3 and 3.2), design & development (Sections 3.3 and 3.4), demonstration (Section 4), evaluation (Section 4 and Section 5), and finally, communication (the paper at hand).

Evaluation of the MuVieMoT approach is performed by using the method in a real-world situation, forcing it to proof its utility and usefulness. Hence, an illustrative scenario (Peffers et al., 2012) is used to showcase the application of MuVieMoT. This evaluation technique comprises to the action research technique as proposed by Siau et al. (2011) for systems analysis and design modeling methods. Besides, the identified requirements are collated with the functionality of the MuVieMoT method in a SWOT analysis.

Requirements towards a Modeling Method for Multi-View Modeling Tools

Before a modeling method is to be designed, generic requirements of the application domain must be gathered. The requirements elicitation is based on three pillars: First, a literature review was conducted in order to gain an overview of different application domains and usage scenarios of MVMMs. Second, the ADOxx meta modeling platform has been investigated in order to derive generic requirements from the modeling tool development domain. More precisely, developing several multi-view modeling tools on ADOxx (see e.g., Bork and Sinz, 2010; Bork and Karagiannis, 2014) revealed shortcomings when it comes to the specifics of MVMMs. Third, the experience of using several multi-view modeling tools influenced the requirements specification by comprising a modeler's perspective.

In total, three scenarios have been identified; delimiting the different purposes of using a conceptual modeling language in the domain of MVMMs:

- **Specification of Multi-View Modeling Language**

Considering the modeling language specification of the viewpoints from a meta modeling perspective, **projective** MVMMs can be distinguished from **selective** (Cicchetti et al., 2011): In selective MVMMs, each viewpoint is implemented as a distinct meta model and the overall system is obtained as synthesis of the different viewpoints. By contrast, end-users in projective MVMMs are provided with virtual viewpoints made up of selected concepts coming from a single base meta model by hiding concepts not relevant for the particular viewpoint. Whereas projective MVMMs already define **syntactic consistency** constraints, i.e., the overlapping viewpoint meta models, selective MVMMs abandon the specification of all syntactic consistency constraints to the method engineer. **Semantic consistency** constraints, i.e., constraints that cannot be automatically derived from the meta models, must be specified in both approaches. Multi-Perspective Enterprise Modeling (Frank, 1994) and the Semantic Object Model (Ferstl and Sinz, 2013) methods are adopting the projective approach, whereas the Orthographic Software Modeling (OSM) (Atkinson et al., 2013) approach creates a single-underlying model for a set of loosely coupled meta models.

The **consistency constraints** between viewpoints are complex. Typically, not even the majority of constraints can be classified as 1:1 copies of modeling concepts, part of several viewpoints. A method should therefore enable consistency constraint specification (e.g., only selected attributes of the same concept, or different attributes of different concepts). Their specification should be supported by a conceptual model, not by a programming language.

Also, **viewpoint-specific visualization** of the modeling concepts should be possible, allowing to adjust the notation to the specific needs of the viewpoints' stakeholders (cf. Bobrik et al. (2007)).

- **Specification of Multi-View Modeling Procedure**

A conceptual modeling language should cover the dynamic aspects of multi-view modeling (cf. Figure 1). Therefore, modeling operations should be aligned to certain viewpoints and the effects of their execution on all other viewpoints should be specified. Persson et al. (2013) identified several process-related relationships between viewpoints, e.g., precedence and (co-)dependency. Such dependencies, resulting in the necessity of providing either *viewpoint transformations* or *viewpoint synchronization* functionality (cf. Figure 1), should be reflected in a conceptual modeling language. Grundy et al. (2013) and Atkinson et al. (2013) emphasized on supporting method engineers in defining navigation between viewpoints; realized by hierarchical relationships between them.

- **Specification of Multi-View Modeling Tools**

The conceptual models should allow for transformation into technical implementations of modeling tools based on a certain tool development environment, e.g., meta modeling platform. Syntactic viewpoint

dependencies should be computed automatically and extended by semantic dependencies. These dependencies should be transformed into synchronization mechanisms, realized on the development environment. Moreover, reuse should be supported to increase efficiency.

The MuVieMoT Modeling Procedure

In the following, the MuVieMoT method is introduced. Figure 2 illustrates the ideological procedure of utilizing the method by means of an iterative approach. Hence, each evaluation might trigger an additional iteration.

Figure 2. MuVieMoT Modeling Procedure

1. Modeling Scenario

The creation of a modeling scenario model is aiming to provide a compact and comprehensive overview of the multi-view modeling method at hand. It centers the modeler and relates him to the model; the viewpoints on that model; the meta models, depicting the syntax and semantics of the viewpoints; the goals he pursues with the model; the metaphor he is guided by; and the relevant subarea of the real world to be mapped (cf. Bork and Sinz, 2013). The focus of the method engineer should be on *identifying the viewpoints*. Additionally, the projective or selective *relationship between viewpoints* and meta models is defined.

2. Modeling Language

The modeling language of the viewpoints is formally specified by meta models. These viewpoint meta models are defined by a selection or a projection on the present meta model(s) - depending on whether or not a common meta model is given or several meta models are loosely coupled. Hence, the meta models should be specified first. Afterwards, viewpoints can be specified by selecting modeling concepts and relating them to each other. Cardinality constraints, inheritance relationships, and abstract classes can be added as required. Moreover, viewpoint-specific properties of the concept can be defined, e.g. a viewpoint-specific visualization.

3. Modeling Procedure

The interaction between a modeler and a multi-view modeling tool is specified by means of *multi-view modeling use cases* (Bork and Sinz, 2013). Conventional UML use case diagrams are adopted by ignoring the actor and relating all use cases to viewpoints by means of *triggers*, *effects*, or *conditional_effects* relationships. The method engineer needs to distinguish for each use case, whether it can be triggered within a certain viewpoint and which effect the execution has on all other viewpoints. Conditional effects are effects that only take place if e.g., a certain state of the view is present, or if optional views exist. Conventional relationships, defining the interplay between use cases, e.g., *include*, *extend*, and *uses*, can be applied also.

4. Viewpoint Dependencies

The viewpoint dependencies model provides an intuitive visualization of *consistency constraints* by relating modeling concepts (i.e., modeling and relation classes) to the viewpoints they are considered in. Moreover, the method engineer specifies whether the concept is used 1:1 in each viewpoint or if only certain attributes of a concept in one viewpoint are kept consistent with certain attributes of that - or a different - concept in a different viewpoint. For example, changing the name of a concept in one viewpoint might result in changing some reference attribute value of a different concept in a different viewpoint.

5. Conceptual Design

The fifth step uses the information gathered in the preceding steps in a comprehensive conceptual design for a multi-view modeling tool. It includes functional requirements that are derived from the modeling procedure step and the consistency requirements specified in the viewpoint dependencies step. Non-functional requirements can be added. For each functionality specified, an informal textual description of

its semantics can be given. Modeling operations can be aligned to a specific viewpoint or a modeling concept within a certain viewpoint.

An emphasis of MUVIEMOT is to consider consistency issues as soon as possible. Therefore, the Conceptual Design provides for each requirement the criteria *Effect* and *Consistency*. The former can be used to describe (e.g., using pseudo code notation) the functionality of the operator. The latter is specifically targeting inconsistencies that need to be considered whenever the modeler executes an operator (cf. Figure 1). Especially the conditional_effect relationships of the modeling procedure step and the solutions for ensuring consistency need to be specified precisely.

6. Evaluation

The conceptual design can be evaluated by method engineers, testing for comprehensiveness of the specification; tool developers, testing for feasibility against the functionality of a tool development environment; or modelers, testing for usability of a prototypical implementation. The gained insights might result in an additional iteration.

The MUVIEMOT Modeling Language

Figure 3 illustrates the modeling language of MUVIEMOT by means of meta models. The illustration adopts UML class diagram notation. Selected dependencies between the different MUVIEMOT models are indicated by red dotted lines. The meta models have been created while implementing a modeling tool supporting the MUVIEMOT method on the ADOxx meta modeling platform (Bork and Karagiannis, 2014). Therefore, some components (e.g., Modeling Class, Relation Class) of the meta models depend on the meta meta model of ADOxx[1].

As the rectangles indicate, steps 1 to 5 of the MUVIEMOT modeling procedure have been realized with distinct meta models. Hence, MUVIEMOT itself can be considered a multi-view modeling method. The red lines visualize not only dependencies, i.e., a Viewpoint can only be specified if it has been introduced in the Modeling Scenario in advance, they also indicate transformations between MUVIEMOT steps. For example, when creating a *Multi-View Modeling Use Case* model, all specified viewpoints are automatically introduced by the tool - if the modeler acknowledges - and automatically placed according to a layout algorithm on the left and right border of the modeling canvas. The modeler then only has to create the use cases and relate them to the viewpoints by means of *triggers*, *effects* or *conditionally_effects* relationships.

Table 1 illustrates the coverage of the identified requirements by mapping them to the components of the MUVIEMOT method.

Requirement	Proposed Coverage in MUVIEMOT
Modeling Language	Meta models and viewpoint models can be specified. An emphasis is put on the specification of different types of consistency constraints between viewpoints.
Modeling Procedure	Modeling operations can be constrained by viewpoints: they can be *triggered-in*; their execution has an *effect* on; or a *conditionally_effect* on.
Modeling Tool	MUVIEMOT comes with a supporting modeling tool. The tool allows model-driven specification and model-driven development of multi-view modeling tools on ADOxx. However, the models aim to be generally and platform-independently applicable as a conceptualization specification.

Table 1. Coverage of Requirements in MUVIEMOT

[1] http://www.adoxx.org, last checked: 2015-04-26

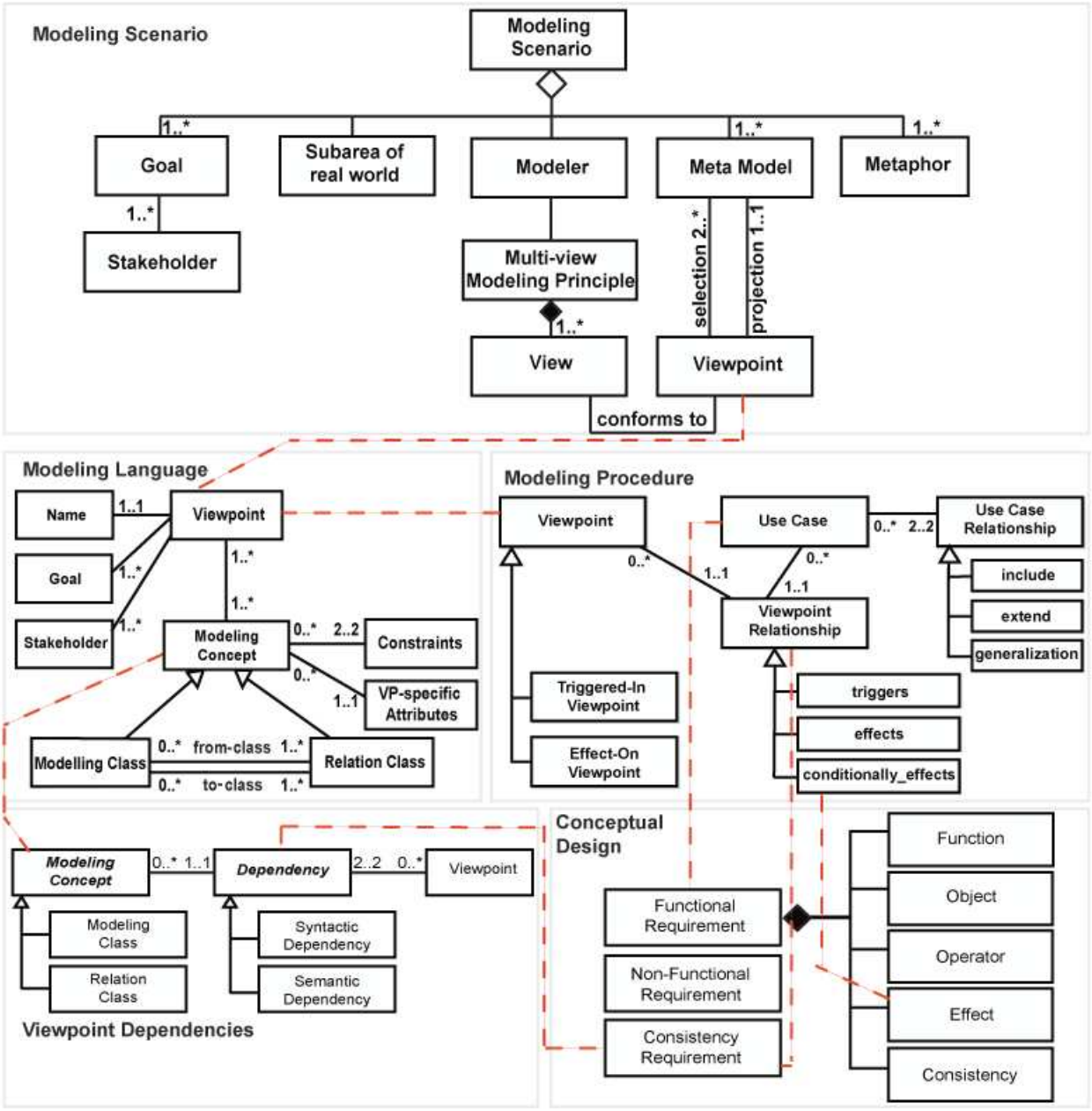

Figure 3. MuVieMoT Meta Models and Relationships

Case Study: Conceptual Design of a SOM Multi-View Modeling Tool

The goal of the case study is to specify a conceptual design for a multi-view modeling tool for the Semantic Object Model (SOM) method (Ferstl and Sinz, 2013). SOM utilizes a multiple perspective approach for analyzing and specifying enterprises. Three perspectives are distinguished: *enterprise plan*, *business process model (bp)*, and *resources*. Modeling of SOM bp models is guided by a multi-view approach, illustrating the coordination of *business objects* by means of *business transactions*. Viewpoints on the structure (**Interaction Schema (IAS)**), the behavior (**Task-Event Schema (TES)**), the decomposition of business objects (**Object Decomposition Schema (ODS)**), and the decomposition of business transactions (**Transactions Decomposition Schema (TDS)**) compose a SOM bp model.

The meta models of the viewpoints are derived by applying a projection operator onto one integrated meta model. A comprehensive introduction to SOM can be found in (Ferstl and Sinz, 2013 p. 196ff). Figure 4

illustrates the bp meta model of the SOM method. TDS and ODS are omitted due to their simplicity. Both solely illustrate hierarchical decompositions of business transactions and business objects using a tree-based visualization of the decomposition protocol.

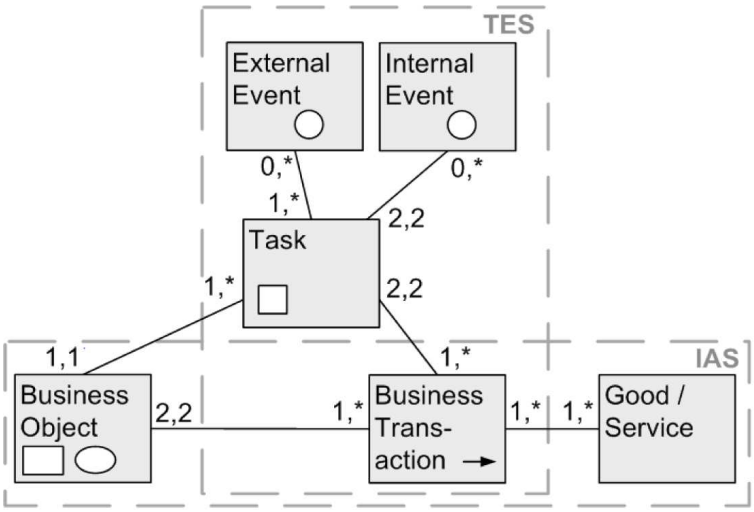

Figure 4. SOM Business Process Meta Model (Ferstl and Sinz, 2013)

Following the procedure of the MUVIEMOT lifecycle, first a **modeling scenario** for SOM bp modeling has been modeled, specifying informally the *metaphor*, the *goals*, the *meta model*, and the *viewpoints*. Figure 5a illustrates the SOM modeling scenario. The models of this case study have been created with the MUVIEMOT modeling tool (Bork and Karagiannis, 2014), developed with ADOxx (Fill and Karagiannis, 2013).

In step 2, the **integrated SOM bp meta model** and the four **viewpoint models** followed (cf. Figure 5c and 5d) - now using a formalized notation (i.e., meta models). MUVIEMOT uses the same notation for the specification of meta models and viewpoint models. Both are visualized in Figure 5b, by means of the **SOM business process modeling meta model** and 5c and 5d, by means of the **Task-Event Schema viewpoint model** and the **Interaction Schema viewpoint model**, respectively.

A meta model is considered platform- and viewpoint-independent, whereas the viewpoint models are considered as realizations of a meta model by means of mapping them to the constituents and functionality provided by a concrete tool development platform. Hence, modeling concepts are specified generic as c**lasses** or **relations**. Figure 5c shows the attribute representation of the Initiating Transaction. Within this representation, the modeler can define the **from**- and **to-endpoints** of a relation, i.e., the classes a specific relation can be connected with. In Figure 5c is exemplified, that *Initiating Transactions* can be connected with *Tasks* as *from- and to-endpoint*.

Editing of SOM bp models is not supported by conventional drag & drop modeling. Instead, SOM defines specific decomposition rules for business objects and business transactions. In the **modeling procedure** step of MUVIEMOT, these modeling operations have been modeled, thereby aligning them to the viewpoints. Subsequently, conditional effects of executing a modeling operation have been specified. One example may illustrate such conditional effects in the context of SOM: many object decomposition rules involve the creation of additional business transactions. Hence, depending on the applied rule, conditional effects on the Transaction Decomposition Schema viewpoint are given. A **viewpoint dependencies** model visualizes the dependencies between modeling concepts and viewpoints.

Finally, functional, non-functional, and consistency requirements have been added to the **conceptual design** model, aiming at the overarching specification of multi-view modeling tool requirements with an emphasis on viewpoint consistency and usability of the tool.

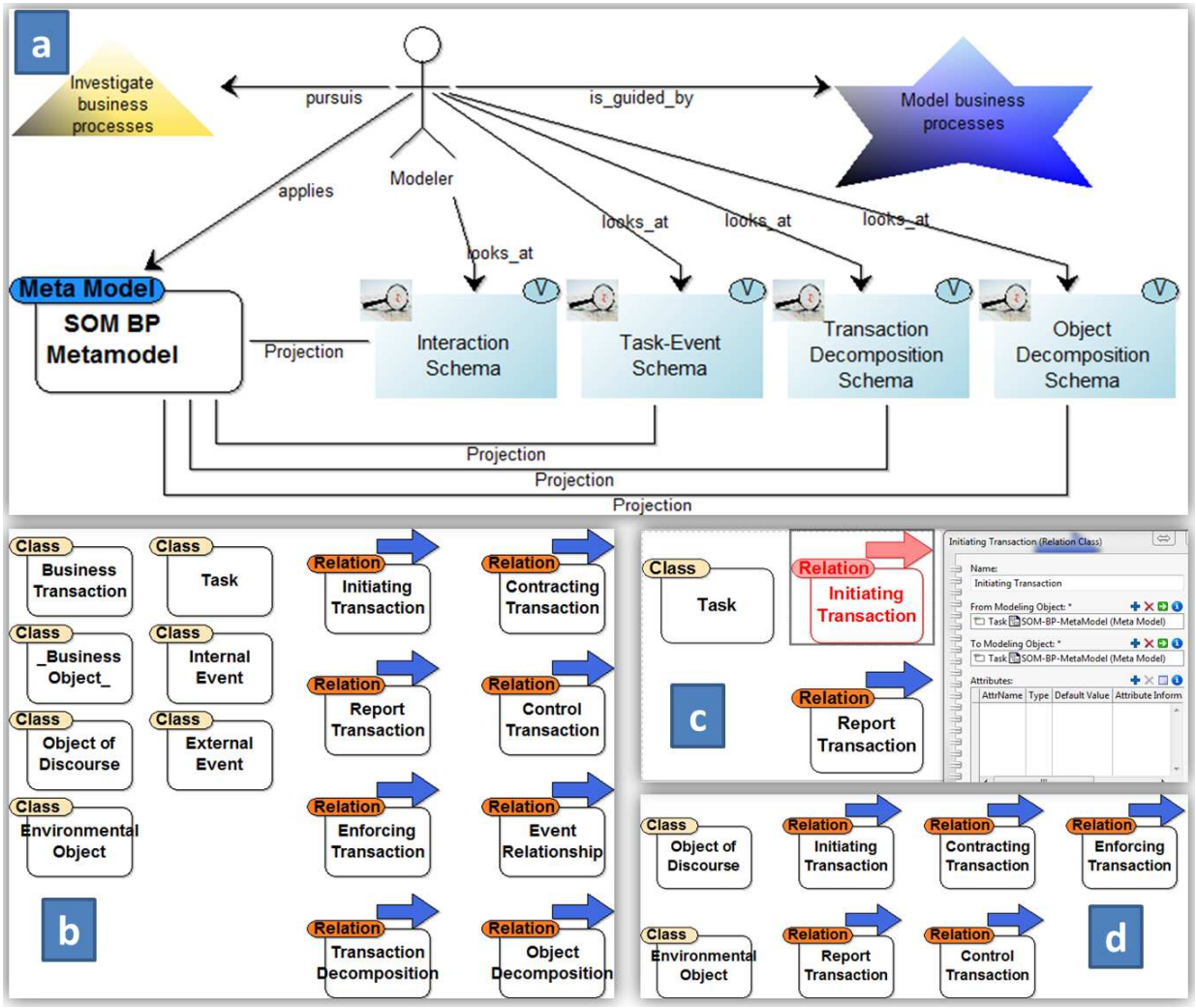

Figure 5. SOM Modeling Scenario (a), Meta Model (b) and Viewpoint models (c, d)

Figure 6 illustrates two more created MuVieMoT models. On the left side, an excerpt of the *multi-view modeling use case* model, created in the modeling procedure step, shows the automatic layout of three viewpoints on the left and right border, and the relationships between viewpoints and use cases. The model visualizes the different notations, implemented to indicate whether the relationship between an use case and a viewpoint is of type *triggers, effects, or conditionally_effects*.

On the right side of Figure 6, the viewpoint dependencies model of MuVieMoT is shown. It visualizes the initially derived syntactic consistency constraints for the concepts *business object* and *business transaction* (blue hexagons). It can be derived, that e.g., the concept *Business Object* is considered in three SOM viewpoints: *Interaction Schema*, *Task-Event Schema*, and *Object Decomposition Schema*. Using the MuVieMoT tool, method engineers can further restrict the consistency constraint by editing the corresponding property in the modeling concepts' attributes (not visualized here).

The whole case study resulted in several models. The interested reader is referred to the MuVieMoT project page[2] for the complete case study. The models eased the transfer of the theoretical specification of the SOM method towards the conceptualization of a SOM multi-view modeling tool.

Comparing this course of action with the experience of developing a SOM tool from scratch (cf. Bork and Sinz, 2010) confirmed two assumptions: (1) guided through the different steps of the MuVieMoT

[2] OMiLAB MuVieMoT project page http://www.omilab.org/web/muviemot/ last checked: 2015-04-26

approach, several consistency constraints have been identified prior to the development and evaluation stages; and (2) using a modeling tool together with generation, transformation, and reuse functionality resulted in a significant increase of efficiency in requirements engineering and tool development.

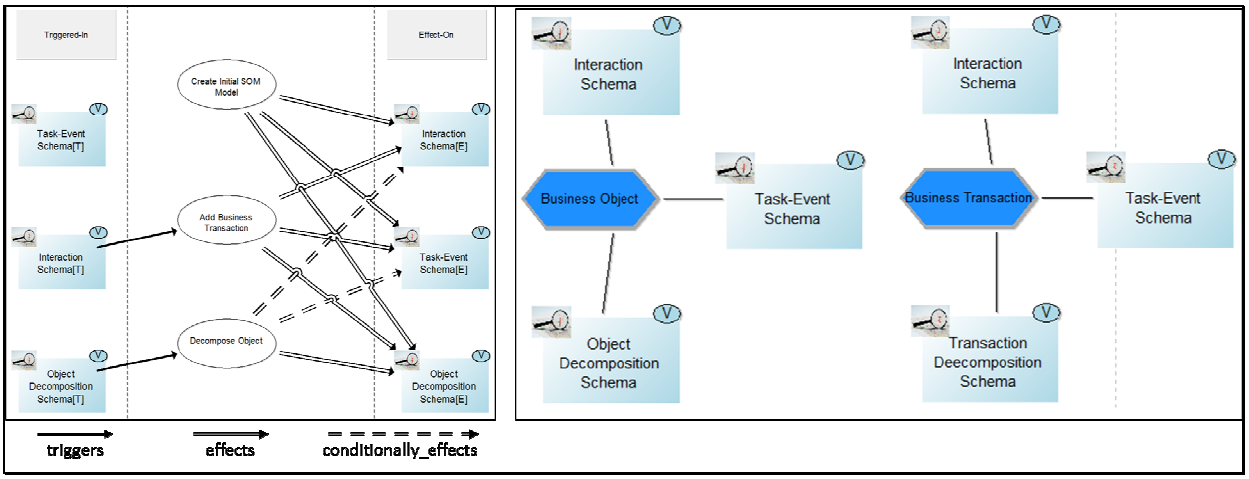

Figure 6. SOM Modeling Procedure (left) and Viewpoint Dependency (right) models

Related Work

Several research groups tried to foster the development of multi-view modeling tools by introducing new specification approaches or tool development environments. The groups come from diverse backgrounds and have different goals, their respective solutions pursue. In the following, these approaches are discussed briefly and contrasted to the MuVieMoT approach.

Atkinson (2013) introduced the Orthographic Software Modeling (OSM) approach. OSM utilizes a single underlying model (SUM) that comprises a set of predefined viewpoints. Interactions are performed solely on the viewpoints and transformed into corresponding edit operations on the SUM. The OSM approach, compared to MuVieMoT, is built on predefined viewpoints that are based on a SUM. Moreover, OSM focuses on software development, whereas MuVieMoT supports the early stages of modeling tool development, i.e., the conceptualization.

Kramer et al. (2013) differentiated themselves from OSM by providing flexible, not predefined viewpoints for modeling software systems. An emphasis is also on correspondences between viewpoints and mechanisms for ensuring consistency. Like for OSM, the focus is on using the multiple views for software development purposes.

Grundy et al. (2013) introduced the Marama meta-toolset, developed on the Eclipse platform. Marama generates multi-view modeling environments following a model-driven approach. The user can specify a single, comprehensive meta model using the *meta-model tool*. Constraints and modeling behavior can be implemented using the Object Constraint Language. A *shape designer* enables the efficient specification of graphical representations for the concepts. Finally, the *view designer* supports the specification of viewpoints by a selection of meta model concepts. In the latest version, Marama also supports the specification of transformations between viewpoints (Grundy et al., 2013). Compared to MuVieMoT, Marama supports the late software development stages utilizing a technical abstraction level, ignoring e.g., requirements engineering. Consequently, the approach is aiming at programmers, not method engineers. Moreover, Marama relies on a single, predefined meta model.

Cicchetti et al. introduced an approach that also emphasizes on the early stages of tool development by providing method engineers guidance in the creation of viewpoints, derived from a common Ecore meta model (Cicchetti et al., 2011, p. 6). Integrated through the meta model, initial syntactic consistency constraints between viewpoints are automatically identified. Also, a multi-view modeling tool based on Eclipse can be generated. The approach is directed towards software development. It focuses on non-graphical models, i.e., the tree-based model editors provided by Eclipse. Moreover, the approach is relying on a predefined single base model, where all viewpoints are projected from.

Kusel et al. (2012) described an approach that combines several modeling viewpoints using a higher abstraction level; realized by one comprehensive ontology model. Using the Web Ontology Language, one is able to define the comprehensive ontology followed by the definition of the viewpoints by means of Eclipse meta models. Integration of and synchronization between the viewpoints is provided by the same domain ontology model. The approach does not consider the early stages properly and omits the modeling procedure completely.

Concluding the review of related works, no consideration of the modeling procedure, i.e., the act of working with multiple views and how to keep them consistent, is provided. Moreover, most of the works utilize a common meta model where all viewpoint meta models must be derived from. The biggest problem, in our opinion, is the lack of an appropriate abstraction level. Most of the approaches are on a technical level and require implementation skills to be utilized. The MuVieMoT method aims to raise the abstraction level towards a more suitable level for method engineers.

Conclusive Remarks

A SWOT analysis has been performed on the current iteration of the MuVieMoT modeling method. The results are based on multiple applications of the method to different multi-view modeling methods. The conclusions are:

Strengths: The presented modeling method supports the conceptualization of multi-view modeling tools by means of conceptual modeling. Hence, it raises the abstraction level and provides method engineers with modeling concepts they are familiar with. Moreover, guided by the step-wise approach, the complexity of modeling tool conceptualization and consistency constraint specification is faced more thoroughly. Although not in the scope of the paper, but using the conceptual models to derive initial modeling tool implementations following a model-driven development approach (Bork and Karagiannis, 2014), significantly reduces implementation effort. Independently of the development approach, the models act as codified knowledge bases, fostering communication and coordination between method engineers and tool developers.

Weaknesses: The MuVieMoT method is strongly aligned to the foundation of modeling methods and meta modeling. If, in the future, more intuitive and efficient paradigms are introduced, the building blocks of the method need to be reconsidered. Moreover, the ADOxx platform (i.e., the ADOxx meta meta model) had obviously some influence on the implementation of the MuVieMoT tool. However, we think that the influence would not have been significantly different if implemented on a different platform like Eclipse. Hence, the resulting conceptual design should be generic and applicable to different tool development platforms.

Opportunities: We are currently working on transforming the conceptual models generated with MuVieMoT automatically into ADOxx tool libraries. These libraries can be imported into the ADOxx platform for automatic derivation of consistency-preserving multi-view modeling tools. An initial transformation is already implemented and has proven its feasibility. We believe, that there are lots of efficiency gains left when providing more code generation and reuse functionality.

Threats: As with any modeling method, the benefit of the created models considerably depends on the knowledge and experience of the modeler. In the case of MuVieMoT, a certain degree of knowledge in meta modeling and multi-view modeling is required in order to create reasonable conceptual designs. The models try to bridge the gap between method owner and tool developer. A serious threat might therefore be that the concepts are considered either to abstract or to technical.

In the paper at hand we introduced the MuVieMoT modeling method. The approach supports method engineers in the conceptualization of multi-view modeling tools by emphasizing on the requirements engineering and conceptualization stages of tool development. The approach has been evaluated using a case study from the enterprise modeling domain, thereby specifying the conceptual design of a multi-view modeling tool for the SOM method.

In the current version, some models are aligned to concepts coming from the ADOxx development platform. However, the MuVieMoT method is intended to capture the generic constituents of MVMMs. Consequently, it is aimed to be generally applicable for MVMMs, fostering the conceptualization of multi-view modeling tools.

REFERENCES

Akehurst, D. H. 2004. "Proposal for a Model Driven Approach to Creating a Tool to Support the RM-ODP," in *Proceedings of the 1st International Workshop on ODP for Enterprise Computing*, pp. 1-4.

Atkinson, C., Gerbig, R., and Tunjic, C. 2013. "A Multi-level Modeling Environment for SUM-based Software Engineering," in *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. *VAO '13*, New York, NY, USA, ACM, pp. 2:1-2:9.

Bobrik, R., Reichert, M., and Bauer, T. 2007. „View-Based Process Visualization," in *Business Process Management, G.* Alonso, P. Dadam, and M. Rosemann (eds.), Volume 4714 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 88-95.

Bork, D. and Sinz, E. J. 2010. "Design of a SOM Business Process Modelling Tool based on the ADOxx Meta-modelling Platform," in *Pre-Proceedings of the 4th International Workshop on Graph-Based Tools (GraBaTs 2010),* J. de Lara, D. Varro, T. Margaria, J. Padberg, and G. Taentzer (eds.), Enschede, The Netherlands, pp. 90-101.

Bork, D., and Sinz, E. J. 2013. "Bridging the Gap from a Multi-View Modelling Method to the Design of a Multi-View Modeling Tool," *Enterprise Modelling and Information Systems Architectures (EMISA) - An International Journal* (8:2), pp. 25-41.

Bork, D., and Fill, H.-G. 2014. "Formal Aspects of Enterprise Modeling Methods: A Comparison Framework," in *Proceedings of the 47th Hawaii International Conference on System Sciences*. *HICSS'2014*, R. H. J. Sprague (eds.), Big Island, Hawaii, USA, IEEE Computer Society Press, pp. 3400-3409.

Bork, D., and Karagiannis, D. 2014. "Model-Driven Development of Multi-View Modeling Tools: The MuVieMoT Approach," in *Proceedings of the 9th International Conference Software Paradigm Trends*, A. Holzinger, J. Cardoso, J. Cordeiro, M. Sinderen, and S. Mellor (eds.), SciTePress, pp. 11-23.

Cicchetti, A., Ciccozzi, F., and Leveque, T. 2011. "A hybrid approach for multi-view modeling," in *Electronic Communication of the ECEASST* (50), pp. 368-377.

Ferstl, O. K., and Sinz, E. J. 2013. *Grundlagen der Wirtschaftsinformatik*, 7th edition, Oldenbourg, München

Fill, H.-G., and Karagiannis, D. 2013. "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform," in *Enterprise Modelling and Information Systems Architectures - An International Journal* (8:1), pp. 4-25.

Fill, H.-G., Redmond, T., and Karagiannis, D. 2012. "FDMM: A Formalism for Describing ADOxx Meta Models and Models," in *Proceedings of International Conference on Enterprise Information Systems*, L. Maciaszek, A. Cuzzocrea, and J. Cordeiro (eds.), SciTePress, pp. 133-144

Frank, U. 1994. "MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems," in *Technology of Object-Oriented Languages and Systems*, R. Ege, M. Singh, and B. Meyer (eds.), pp. 367-380.

Frank, U., Strecker, S., Fettke, P., vom Brocke, J., Becker, J., and Sinz, E. J. 2014. "The Research Field "Modeling Business Information Systems"," in *Business & Information Systems Engineering*, (6:1), pp. 1-5.

Grundy, J., Hosking, J., Li, K. N., Ali, N. M., Huh, J., and Li, R. L. 2013. "Generating Domain-Specific Visual Language Tools from Abstract Visual Specifications," in *IEEE Transactions on Software Engineering*, (39:4), pp. 487-515.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," in *MIS Quaterly,* (28:1), pp. 75-105.

Karagiannis, D., and Kühn, H. 2002. "Metamodeling Platforms," in *Third International Conference EC-Web 2002 – Dexa 2002*, K. Bauknecht, A. Min Tjoa and G. Quirchmayr (eds.), Aix-en-Provence, France, Springer-Verlag, Berlin, Heidelberg, pp. 182-195.

Kheir, A., Naja, H., Oussalah, M., and Tout, K. 2013. "Overview of an Approach Describing Multi-views/ Multi-abstraction Levels Software Architecture," in *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, L. A. Maciaszek, and J. Filipe (eds.), SciTePress, pp. 140-148.

Kramer, M. E., Burger, E., and Langhammer, M. 2013. "View-centric Engineering with Synchronized Heterogeneous Models," in *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, VAO '13, New York, NY, USA, ACM, pp. 5:1-5:6.

Kusel, A., Mitsch, S., Retschitzegger, W., Schwinger, W., Mayr, R., and Schönböck, J. 2012. „Ontology-Driven Generation of Multi-View Modeling Tools," in *Proceedings of 11th IASTED International Conference on Software Engineering*, SE 2012, ACTA Press, pp. 45-51.

Marincic, J., Mader, A., Wieringa, R., and Lucas, Y. 2013. "Reusing knowledge in embedded system modelling," in *Expert Systems*, (30:3), pp. 185-199.

Mylopoulos, J. 1982. "Conceptual Modelling and Telos," in *Conceptual Modeling, Databases and Case: An Integrated View of Information Systems Development*, P. Loucopoulos, and R. Zicari (eds.), John Wiley & Sons, pp. 49-68.

Nuseibeh, B., Kramer, J., and Finkelstein, A. 1994. „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," in *IEEE Transactions on Software Engineering*, (20:10), pp. 760-773.

Peffers, K., Rothenberger, M., Tuunanen, T., and Vaezi, R. 2012. "Design Science Research Evaluation," in *Design Science Research in Information Systems. Advances in Theory and Practice*, K. Peffers, M. Rothenberger, and B. Kuechler (eds.), Volume 7286 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 398-410.

Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. 2007. "A Design Science Research Methodology for Information Systems Research," in *Journal of Management Information Systems*, (24:3), pp. 45-77.

Persson, M., Torngren, M., Qamar, A., Westman, J., Biehl, M., Tripakis, S., Vangheluwe, H., and Denil, J. 2013. "A Characterization of Integrated Multi-View Modeling in the Context of Embedded and Cyber-Physical Systems," in *Proceedings of the International Conference on Embedded Software (EMSOFT)*, pp. 1-10.

Siau, K., and Rossi, M. 2011. "Evaluation Techniques for Systems Analysis and Design Modelling Methods: A Review and Comparative Analysis," in *Information Systems Journal*, (21:3), pp. 249-268.

Sinz, E. J. 1996. „Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme. Entwicklung, aktueller Stand und Trends," in *Information Engineering. Wirtschaftsinformatik im Schnittpunkt von Wirtschafts-, Sozial- und Ingenieurwissenschaften*, H. Heilmann, L. J. Heinrich, and F. Roithmayr (eds.), Oldenbourg Verlag, München, Wien, pp. 123-143.

Zachman, J. A. 1987. "A Framework for Information Systems Architecture," in *IBM Systems Journal*, (26:3), pp. 276-292.