Model-driven Development of Multi-View Modelling Tools
The MuVieMOT Approach

Domenik Bork and Dimitris Karagiannis
Research Group Knowledge Engineering, University of Vienna, Währinger Street 29, 1090 Vienna, Austria

Keywords: Multi-View Modelling, Meta Modelling, Modelling Tools, Model-driven Development, MuVieMOT, ADOxx.

Abstract: As the complexity of modern computer and enterprise systems is ever increasing due to emerging technologies and the need to integrate different systems, modelling tools, designed to encourage modellers in creating models according to the complex reality are of rising importance. Multi-view modelling methods (MVMM) can cope with this complexity by providing visualization, decomposition, and specialization functionality. The creation of a model is decomposed into the creation of several views and integrating them in order to derive the whole model of the system. Keeping the multiple views consistent and providing suitable visualization means is vital for applicability and usability of MVMMs. By contrast, when designing such tools, one is forced to adopt conventional software engineering approaches. The paper at hand tries to contribute filling that research gap by introducing a model-driven approach, tailored to the specifics of designing multi-view modelling tools. A prototypical implementation of the approach enables automatic generation of modelling tools for MVMM using the ADOxx meta modelling platform.

1 INTRODUCTION

Modelling of enterprise systems and information systems is of increasing complexity due to emerging technologies and standards, increasing heterogeneity and globalization, and the different stakeholders involved in creating or processing of the models. Multi-view modelling methods (MVMM) help to cope with this complexity by dividing the modelled system into multiple views. Each view considers only certain aspects of the system thereby utilizing a certain perspective on it (Bork and Sinz, 2013). The views use the abstraction level and concepts most suitable to map the aspects of the subarea considered by the view onto the model. Consequently, aspects not considered by the view are ignored.

An example from the enterprise modelling domain should help to introduce a common understanding: In enterprise modelling, structural, behavioural, technical and organizational aspects are most commonly present. These aspects are modelled using specialized views. Integrating these views manually into one comprehensive model is likely to overwhelm the modeller. This not only holds for analysis of the model, it also covers the process of creating valid models. Most enterprise modelling methods (e.g., ARIS, MEMO, SOM) therefore divide the model into several, interrelated, partial views. Each view is specified by a viewpoint which depicts the concepts considered by the view and the rules for combining them. The relationship between view and viewpoint is analogous to the relationship between model and meta model.

Using such specialized views enables two major benefits: First, the viewpoints can utilize several abstraction and visualisation levels, thereby fostering analysis and understanding of the model. Second, the concepts of the viewpoint can be aligned to the domain or stakeholder, the view is designed for – cf. the benefits of domain-specific modelling languages (DSMLs).

However, due to the unifying character of the viewpoints, correspondences between the different views are inherently given. The adequate definition and handling of these correspondences and the anticipation of inconsistencies adhering from them is vital for the efficient application of MVMMs.

Specifying these correspondences in a formalised and machine-processable manner is one of the major research issues in the currently emphasized research field of developing multi-view modelling tools (Frank et al. 2014). Formalisation enables not only

Karagiannis D. and Bork D.
Model-Driven Development of Multi-View Modelling Tools: The MUVIEMOT Approach.
DOI: 10.5220/0006811900010001
In *Proceedings of the 9th International Conference on Software Paradigm Trends (ICSOFT 2014)*, pages 11-23
ISBN: 978-989-758-037-6

machine-processing but also intersubjective understanding of the specification (Bork and Fill, 2014). Following our introductory example from the enterprise modelling domain, a behavioural viewpoint (e.g., for business processes) has relations to the technical and organizational viewpoints (e.g., machines and personnel for the execution of certain tasks defined in the business process). The breakdown of a machine is normally captured in the technical view. However, it is obvious that in case of a machine breakdown, business process tasks performed with this machine cannot be performed until the machine is repaired. Such correspondences between views do not only effect the operation of enterprises but also their planning and analysis.

The development of consistency-preserving multi-view modelling tools is therefore a key factor for utility and efficiency of MVMM. Currently, this task is conceptually and technically prominently supported by meta modelling and meta modelling-based tool development platforms like ADOxx (Fill and Karagiannis 2013), Eclipse Modelling Framework, or MetaEdit+ (Tolvanen and Rossi, 2003). These platforms share the generic steps to be undertaken in order to create a modelling tool.

However, such platforms concentrate on the late development steps, i.e., the implementation of modelling tools. Specifying the requirements and the functionality is still left to the cognitive capabilities of method engineers. This is even more a shortcoming when thinking of the specifics of MVMMs. There is a significant lack of support when it comes to the design of multi-view modelling tools (Bork and Sinz, 2013).

The paper at hand aims to reduce this deficit by introducing a set of domain-specific modelling languages (DSMLs) for the model-driven specification of requirements for multi-view modelling tools. The DSMLs and a procedural approach defining their application is in the following referred to as MUVIEMOT. An implementation of the approach on the ADOxx meta modelling platform moreover allows the automatic transformation of the models into modelling tool implementations. The paper describes MUVIEMOT and illustrates its feasibility using a case study from the enterprise modelling domain.

The rest of the paper is organized as follows: Section 2 describes the components of modelling methods and uses these components to further specify multi-view modelling methods. A brief introduction to the development of modelling tools concludes the foundations. In Section 3, the DSMLs for the specification of multi-view modelling tools are introduced. The model-driven engineering of multi-view modelling tools with MUVIEMOT is then described in Section 4. A case study, follows in Section 5. Finally, Section 6 concludes the paper and gives some ideas for further research.

2 FOUNDATIONS

This section outlines the foundations of modelling methods. Subsequently, the specifics of a multi-view modelling method are contrasted. Finally, Section 2.3 briefly describes current development approaches, emphasizing on their inappropriateness for the development of multi-view modelling methods.

2.1 Modelling Methods and Meta Modelling

Due to the increasing complexity of today's enterprises and the software systems determining their ability to compete in a global market, models play a vital role. Models not only help to cope with the complexity by providing structuring, analysis and further processing qualities. They are also used as primary tool for the development of the software systems, e.g., by following a model-driven development (MDD) approach.

A modelling method, according to (Karagiannis and Kühn, 2002) is composed of three major parts: (1) a *Modelling Language*, (2) a *Modelling Procedure*, and (3) *Mechanisms & Algorithms*. Figure 1 illustrates the components of a modelling method by means of an UML class diagram.

A *Modelling Language* defines the elements of the modelling method and the rules for combining them. For every element, semantics defining their meaning, and notation, defining their graphical visualization need to be specified.

The *Modelling Procedure* then uses the specified modelling language and defines the steps and results while actually creating valid models by a modeller. The combination of the modelling language and the modelling procedure is referred to as the *Modelling Technique*.

Mechanisms & Algorithms "provide the functionality to use and evaluate the models built by using the modelling language" (Karagiannis and Kühn, 2002, p. 4). According to the meta model level the mechanisms are specified on, three different classes of mechanisms can be identified. *Generic* mechanisms are specified on the meta meta model. They can therefore be applied on every meta

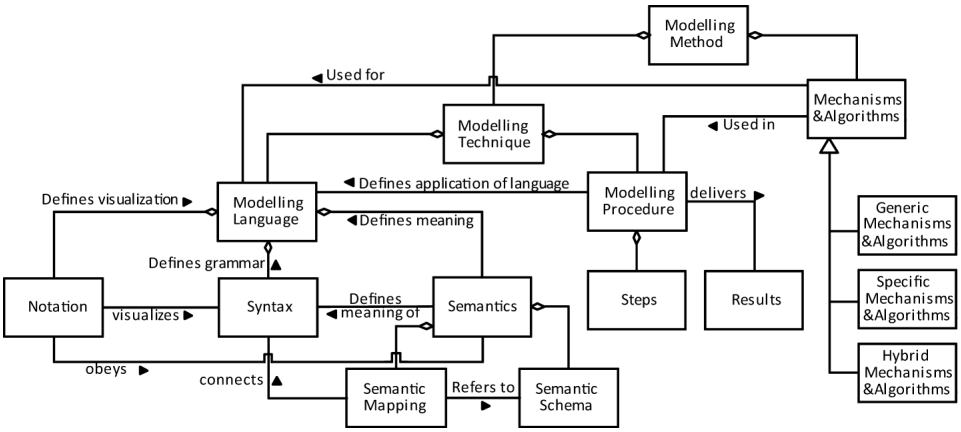

Figure 1: Components of modelling methods (Karagiannis and Kühn 2002).

model defined using the concepts of the meta meta model. *Specific* mechanisms on the other side are implemented using the concepts of a certain meta model. Finally, *Hybrid* mechanisms are defined on the meta meta model but they can be adopted or parameterized using the specific concepts of the meta model, e.g. customization of pre-defined analysis queries.

Based on the definitions of modelling methods, we now describe our understanding of meta models and meta modelling. In contrast to *modelling*, where the application of a modelling method is centred, i.e. the creation of a model as an instance according to a meta model, *meta modelling* is concerned with the formalized specification of a modelling method. This specification covers all components of Figure 1.

In this context, meta models are generally referred to as *"a model used to model modelling itself"* (Fill and Karagiannis, 2013, p. 6f). In more detail, a meta model defines the abstract syntax of a modelling language (Harel and Rumpe, 2000; Harel and Rumpe, 2004; Sprinkle and Rumpe, 2010). Accordingly, meta meta models then define the language that is used to describe this abstract syntax. Whereas in some approaches separate meta models are used for distinct domains, e.g. in the area of model transformation by using meta models for software engineering and meta models for database modelling (Romeikat et al., 2008), other approaches use integrated views on one meta model whereby domains are distinguished by using different model types (Fill et al., 2012). These model types can be referred as viewpoints on the model.

2.2 Multi-View Modelling Methods

During the design and analysis of complex information systems (e.g., enterprise information systems, knowledge systems) multiple aspects need to be taken into account simultaneously (Alter, 2008). This includes information about the subject, usage, systems, and development world of these systems. By reverting to multi-view modelling approaches, multiple structural and behavioural aspects can be represented using different, inter-connected modelling languages (i.e., viewpoints). Historically, the multi-view modelling approach can be aligned to the database engineering domain. In relational databases, a view is a non-materialized subset of the attributes of a database table, derived by a projection or selection operator. Over the last decades, MVMMs have been successfully applied in different domains like requirements engineering (Finkelstein and Fuks, 1989; Finkelstein et al., 1992), architecture management (Kruchten, 1995), software modelling (Dijkman et al., 2008; Nassar, 2003), or software development (Mili et al., 1999) A more comprehensive overview on the application domains of viewpoint modelling can be seen in (Kheir et al., 2013).

Multi-view modelling methods help to cope with the complex reality by providing specialized viewpoints. Each view is focusing only on certain aspects of the reality, therefore enabling a highlighting of the considered aspects by explicitly omitting other aspects. The combination of the views gives the whole model of the system. As a consequence of the interrelated views, correspondences need to be specified in order to prevent inconsistencies (cf. the discussions on the view-update problem in database engineering (Carlson and Arora, 1979; Dayal and Bernstein, 1978; Dayal and Bernstein, 1982; Lechtenbörger, 2003).

Generally, two oppositional categories of multi-view modelling methods can be identified: *selective and projective multi-view modelling* (Cicchetti et al., 2011):

- **Selective:** Each viewpoint is implemented as a distinct meta model and the overall system is obtained as synthesis of the information carried by the different viewpoints.
- **Projective:** Modellers are provided with virtual views made up of selected concepts coming from a single base meta model by hiding details not relevant for the particular viewpoint.

From a meta modelling perspective the two categories can also be classified by either providing the combination of several loosely coupled meta models (i.e., selective MVM), or by the availability of one single, integrated meta model the views are derived from by a projection operator (i.e., projective MVM). Whereas the integrated meta model of the former category already defines some of the consistency constraints a supporting modelling tool should respect, the second category abandons the specification of all consistency constraints to the method engineer. Multi-Perspective Enterprise Modelling (MEMO) (Frank, 1994) and the Semantic Object Model (SOM) (Ferstl and Sinz, 2013) are two modelling methods from the enterprise modelling domain adopting the projective approach, whereas the Orthographic Software Modelling (OSM) approach creates a *single-underlying model (SUM)* for a set of loosely coupled meta models (Atkinson et al., 2013) in the software engineering domain.

Multi-view modelling methods do not only require specific thoughts considering the definition and integration of the views, i.e., the specification of the multi-view modelling language by means of a combination of meta models. They also effect the specification of the other two major components of modelling methods, the modelling procedure and the mechanisms & algorithms.

The actions a modeller can perform must be specifically aligned to the characteristics of MVMMs. For each modelling action, it should be decided in which viewpoints they can be triggered and on which viewpoints the execution of an action has an effect on. Bork and Sinz proposed two categories of performing multi-view modelling, referred to as *multi-view modelling principles*: system-oriented and diagram-oriented multi-view modelling (cf. Bork and Sinz, 2013). In the former case, *an operator is applied to a specific diagram or to the model itself. Its effects can be seen in all corresponding diagrams"* (Bork and Sinz, 2013). The modelling tool automatically ensures a consistent model state for all views automatically. In order to do this, all correspondences between the viewpoints must be specified formally. In the latter case, *"editing of a model is done by applying an operator to a single diagram. The effects of the operator can only be seen in the diagram" (*Bork and Sinz, 2013). The modelling tool explicitly allows temporarily inconsistencies between the views, i.e., due to usability. The modeller must perform additional actions in related views in order to obtain a consistent model state for all views. Figure 2 illustrates the dependencies between views in a multi-view modelling setting.

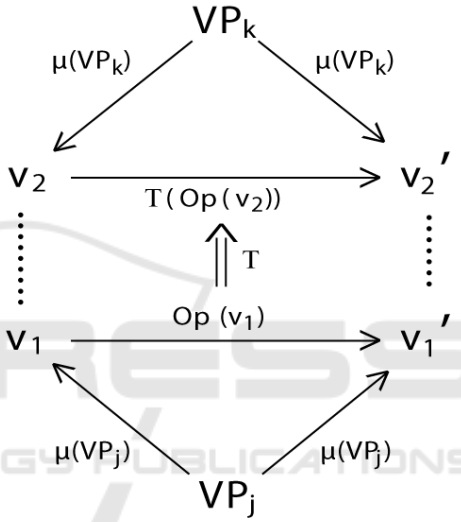

Figure 2: View dependencies in multi-view modelling.

Figure 2 illustrates the interplay of viewpoints, correspondences, and meta models using a conceptual model (the visualization is an analogy to the view-update problem, described in relational databases in the late 1980s (Keller, 1985).

Compared to models, which are instances of a meta model, the constituents of a view are specified by a Viewpoint (VP). View v_1 is an instantiation of the Viewpoint VP_j, View v_2 is of type VP_k. The application of an operator (Op) on v_1 transforms view v_1 into a new model state, referred to as v_1'. A modelling tool supporting system-oriented multi-view modelling should provide a transformation of operator Op (T(Op)) that can be applied on view v_2, transforming the view into the new state v_2'. The overall goal of a consistency-preserving multi-view modelling tool is highlighted by the dotted lines between v_1 and v_2, and v_1' and v_2', respectively. These lines indicate a semantic consistency relationship

between the views, i.e., the information covered in the views does not contradict each other.

Considering mechanisms & algorithms of a multi-view modelling method, additional thoughts must be given e.g., to the view-specific visualization of concepts that are included in several viewpoints. A tool development environment should provide the possibility to define a view-dependent visualization in order to visualize the concepts most appropriately to the semantic domain the viewpoint is defined for.

2.3 Development of Modelling Tools

Although the benefits of using modelling methods are clearly documented in the research community, methodical support in the early phases of modelling tool development is rarely given.

Currently, the support of a method engineer, trying to implement a modelling tool supporting his or her method, is basically the technical specification of tool development environments. The development of modelling tools is, up to now, seen as a special kind of conventional software engineering. Meta modelling platforms have enabled a more sufficient support in the late development steps, however, they lack at support for the requirements engineering and conception of modelling tools.

Traditional software development procedure models like the *Waterfall Model* (Benington, 1983) or the newer approaches subsumed under the term *agile software development* however cannot consider the specifics of MVMM appropriately. Due to the even more challenging characteristics of multi-view modelling methods (cf. Section 2.2) the gap in methodical support is even more serious. MUVIEMOT, proposed in the following Section, and the underlying methodical approach contribute to close that research gap.

The concepts of meta models and meta modelling do not only provide abstraction mechanisms for human beings, they are also commonly used as conceptual foundation for tool development platforms. Such platforms often provide a fixed meta meta model (Sprinkle et al., 2010). Tool developers then integrate the meta model of their modelling method by using the concepts provided in the platform's meta meta model. The platform generates the visual graphical editor based on the meta model and the notation defined for its elements. This enables modellers to create models according to the meta model.

The ADOxx meta modelling platform follows the same procedure. ADOxx has emerged from Adonis, a business process management tool which is in commercial use until now. A free of use version of ADOxx is available for academic purposes. This version can be used to easily develop a graphical modelling tool using the meta modelling concept. Figure 3 illustrates the roles and languages of the ADOxx meta modelling hierarchy.

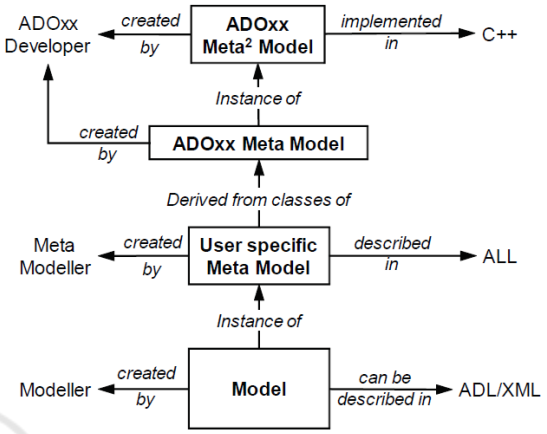

Figure 3: Roles and languages of the ADOxx meta modelling hierarchy (Fill and Karagiannis, 2013).

On top of the hierarchy is the ADOxx meta meta model which is created by the ADOxx developers and implemented using the programming language C++. As an instance of this meta meta model, the ADOxx meta model has been created. This meta model can be used by meta modellers to create their own meta model. The creation is performed by mapping the user-specific meta model elements to the elements provided within the ADOxx meta model. The resulting meta model is described in the ADOxx Library Language (ALL), a platform-specific language for describing meta models using the concepts of the ADOxx meta meta model. Finally, the modeller can create instances of the user-specific meta model, i.e. create actual models. These models can be described using the pre-defined standard export formats XML or ADL. Adonis Description Language (ADL) is a ADOxx-specific XML format.

The mapping of the concepts between the user-specific meta model and the ADOxx meta model is now explained in more detail. Central concepts of an ADOxx library are *Modeltypes*, *Modelling Classes*, *Relation Classes*, and *Attributes* (cf. Junginger et al., 2000). Modelling and Relation Classes are composed of attributes. Some of them are already mandatory by the platform, e.g. the *GraphRep* attribute for the definition of the graphical representation or the *AttrRep* attribute for the

visualization of an object's attributes by means of an ADOxx *Notebook*. Other attributes can be introduced by the method engineer. For each Relation Class *from* and *to* which Modelling Classes the relation can be modelled must be specified. A modeltype in ADOxx delimits which Modelling Classes and Relation Classes should be included within a certain model in the ADOxx Modelling Toolkit.

3 MUVIEMOT: A DSML FOR MULTI-VIEW MODELLING TOOLS

Domain-specific languages (DSLs) enable the declaration of concepts that describe the intended usage most appropriately. The resulting language is precisely tailored to a certain domain, fulfilling a pre-defined purpose. By definition, this is something general-purpose languages cannot provide. DSLs are widely used in software development projects in a diverse set of domains.

Following the definition of DSL, a domain-specific modelling language (DSML) can be defined as *"a modelling language that is intended to be used in a certain domain of discourse. It enriches generic modelling concepts with concepts that were reconstructed from technical terms used in the respective domain of discourse. A DSML serves to create conceptual models of the domain, it is related to"* (Frank, 2010, p. 4). Moreover, DSMLs enable the model-driven development of systems by providing a higher abstraction level (Tolvanen, 2005).

3.1 Requirements on a DSML for Multi-View Modelling Methods

Before we describe the MUVIEMOT approach, we first briefly discuss some requirements a requirements modelling for multi-view modelling methods should adhere. From a functional point of view, a DSML for MVMM should be able to capture all facets of the methods. This includes the meta models, the viewpoint definitions, the consistency constraints, and the visualization of the views. Moreover, an emphasis of the DSML should be on defining the modelling procedure of the MVMM as actions performed by the modeller on a certain view (or a set of views) and the effects of these actions (cf. Figure 2).

Visualization mechanisms play an important role for conventional modelling methods. This all the more holds for MVMM, as concepts may be included in different views using different visualization paradigms and notations.

From a non-functional perspective, the DSML should follow its originating purpose, i.e. provision of an abstraction level and concepts that are strongly aligned to the domain. In the case of the MUVIEMOT approach, it is important, to use the concepts provided with the foundations of modelling, meta modelling and multi-view modelling. Classical non-functional requirements like usability, scalability, and robustness should be also considered.

3.2 Designing Multi-View Modelling Tools with MUVIEMOT

The aim of MUVIEMOT is to increase the efficiency of meta model based specification and model-driven development of multi-view modelling tools by providing a more suitable level of abstraction. This abstraction level enables the specification of the MVMM in a more efficient way due to the provided modelling concepts who are strictly aligned to the domain and the needs of method engineers. Central models of the DSML focus on the overall setting of the MVMM, the use cases of applying the MVMM tool, and the consistency issues a tool developer should consider during implementation of the tool. Therefore, the models help to gather all requirements of a multi-view modelling tool

In the following, the phases of the procedural approach, the DSML is based on, are described generally (cf. Bork and Sinz, 2013). Afterwards the realization of the phases in a supporting modelling tool is described. The MUVIEMOT tool is implemented on the ADOxx meta modelling platform using the facilities of the Open Models Initiative (Karagiannis et al., 2008) laboratory (OMiLAB). Although MUVIEMOT is developed on ADOxx and the transformation only supports the ADOxx Development environment, the phases are generically applicable in the early steps of modelling tool development, independently of the development platform used.

3.2.1 Modelling Scenario

Due to the complex setting of multi-view modelling methods (cf. Section 2.2), the first phase in the procedural approach is trying to obtain an overview over this complex setting. The *Modelling Scenario*

model is therefore directed towards supporting method engineers in defining the overall setting of the multi-view modelling method. This setting includes the *goals* that can be derived by the stakeholders, the *metaphor* the modeller should be guided by and of course the *viewpoints* and the *meta models* they are derived from. Additionally, the *sub-area* of the real world covered by the model should be delimited.

Most of these aspects can be specified informally, e.g., using natural language. Considering the meta models, the viewpoints, and the relations between meta models and viewpoints, a formalized specification enables machine-processing of the models. The meta models and the viewpoints are both specified using the modelling concepts of ADOxx. Each concept of the meta model is therefore modelled as either *Modelling Class* or *Relation Class*. MUVIEMOT enables the specification of the *notation*, the representation of the object instance's *attributes* (i.e., a ADOxx Notebook), and *user-specific attributes*. Moreover, inheritance relationships between Modelling Classes can be specified. The result of the Modelling Scenario is a complete specification of all viewpoints and meta models together with contextual information.

3.2.2 Multi-View Modelling Use Cases

The second phase uses the Modelling Scenario for the specification of *Multi-View Modelling Use Cases* (in the following referred to as *use case*). Each use case depicts a modelling action, realized by an interaction between the modeller and a viewpoint of the modelling tool.

For each use case the method engineer can depict in which viewpoints it can be *triggered* and on which the execution of the use case has an *effect* on. Moreover, relationships between use cases can be defined (e.g., include, extend). The approach differentiates between a *direct effect*, *no effect*, and a *conditional effect* (e.g., the execution of an use case in viewpoint A has only an effect on viewoint B, if B contains a certain concept c). The different relationship types are graphically visualized differently, allowing immediate interpretation of their semantics.

It is important to note, that the relationships between different viewpoints are very complex. Not all relationships are bi-directional, i.e., depending on which view triggers a change, edit operations on other views are performed or not. Another important aspect is the fact, that the concepts who have to be consistent do not always have to be same (e.g., they have different semantics, only selected attributes are kept consistent). Experience in the development of multi-view modelling tools showed, that most of the relationships are very complex. Often, changing of attributes in one view results in a set of temporarily inconsistencies that needs to be resolved by the tool developer. Therefore, a model-based approach for defining these dependencies is very useful.

3.2.3 Conceptual Design

The third phase of the approach considers the information gathered in the preceding two steps and combines them to a conceptual design specification of a multi-view modelling tool. The conceptual design provides all functional requirements derived from the second step supplemented with non-functional requirements for a MVM tool.

One emphasis of the conceptual design is on the consistency between the views. Therefore, especially the conditional effects defined in the multi-view modelling use cases must be specified thoroughly. Consequently, these consistency requirements enable a tool developer for a more efficient implementation. As for conventional software engineering, early conception and design mistakes are very expensive if they are revealed later. Concentrating on these early steps in the conception of a multi-view modelling tool should result in a more mature and consistency-preserving modelling tool.

The Conceptual Design model allows the specification of the requirements in two ways, either graphical or tabular. First, all functional requirements derived from the second step, the multi-view modelling use cases, are included. Then, the method engineer can further define functional and non-functional requirements. The constituents of a requirement in the Conceptual Design model are function, object, operator, effect, and consistency (see (Bork and Sinz, 2013, p. 8f) for a comprehensive description of the constituents).

4 MODEL-DRIVEN DEVELOPMENT OF MULTI-VIEW MODELLING TOOLS

The MUVIEMOT approach not only supports modelling, specification, analysis and documentation of MVM tools (cf. Section 3). The generated models can be also used to derive a multi-view modelling tool following a model-driven development (MDD) approach. MDD *"technologies*

offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively" (Schmidt, 2006).

The starting point for the transformation is the Modelling Scenario model. It includes the specification of the meta models and the viewpoints. Viewpoints are realized as model types in ADOxx.

In the following, the algorithm for transforming the Modelling Scenario model into ADOxx Library Language (ALL) code is explained.

4.1 Transforming the Modelling Scenario into ALL Code

First, the algorithm searches for all *Viewpoint* objects within the Modelling Scenario model. Each view is being transformed into a *MODELTYPE* (cf. Section 2.3) specification in the resulting ADOxx library. A modeltype depicts the modelling classes and relation classes that are considered within a model in ADOxx. In order to delimit these constituents, each Viewpoint is referencing a Viewpoint Model. Within this model, all concepts considered by the Viewpoint are defined by projecting or selecting them from the specified meta models. The algorithm includes all classes into the ADOxx MODELTYPE, together with some modeltype attributes the ADOxx platform supports.

After all classes are included, the actual meta model needs to be specified using the concepts provided by ADOxx. Therefore, the Meta Model model referenced in the Modelling Scenario model is retrieved. Within this meta model all classes are specified comprehensively, including all user-defined attributes and the attributes required by ADOxx (e.g. the graphical representation and the definition of the Notebook). If an inheritance relationship is defined for a modelling class, all attributes specified in the super class are also included in the sub class. All information is gathered and appended to the ALL specification of the library.

The generated ALL specification allows the immediate construction of an initial modelling tool for the designed (i.e., modelled) MVMM. Depending on the complexity of the method and the constraints for building models according to the method, additional implementation needs to be performed. This implementation essentially regards the modelling procedure and the mechanisms & algorithms of the method. Consequently, the generation limits the effort for implementation to a minimal level. At some points during the modelling process, the method engineer might be already aware of functionality the developer needs to implement on the platform (e.g., considering the storage of the models or import/export format of the models). In such cases, the MUVIEMOT tool also provides the method engineer with the possibility to define some informal or pseudo-code specification of the functionality easing the transfer of the requirements between method engineer and tool developer. Moreover, including the Multi-View Modelling Use Cases and the Conceptual Design to the MUVIEMOT tool in the future will close this gap.

4.2 View Consistency Mechanisms

MUVIEMOT not only transforms the Modeling Scenario into ADOxx modeltypes, the tool is also able to compute a *View Dependency* model. The algorithm searches in all Viewpoint Models for the included concepts, i.e., modelling and relation classes. For each concept, a list of views that include the concept is generated. After all models are checked, a new model is created that visualizes the dependencies between view concepts and views.

The transformation algorithm then uses this View Dependency model in order to add the concepts that should be kept consistent when changes are performed by the modeller in a certain view. This generated synchronization mechanism is executed whenever the modeller performs a modelling action on the platform. If the changes affect a concept that should be kept consistent, the propagation of the changes to all corresponding views is automatically triggered.

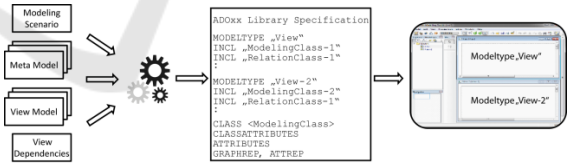

Figure 4: MUVIEMOT transformation process.

Figure 4 illustrates the transformation process by highlighting the different MUVIEMOT models and the information retrieved from them during generation of the ADOxx library description in ALL code. The ALL code can then be converted into an actual ADOxx library (referred to in the following as abl). A web converter implemented and operated by the BOC allows the seamless conversion of ALL code into an abl file. This file can then be imported into the Development Toolkit of ADOxx. Within seconds, the platform integrates the new library and enables immediate creation of multi-view models

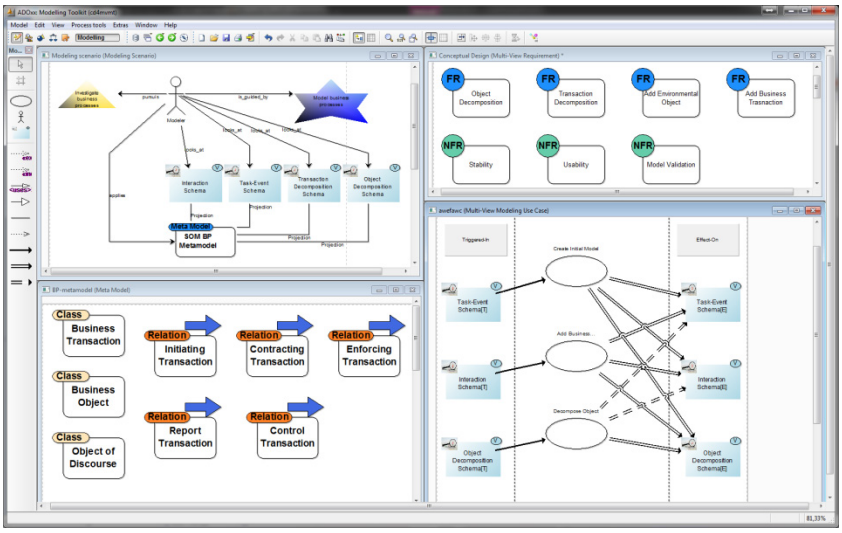

Figure 5: MUVIEMOT models for the SOM bp modelling method.

using the Modelling Toolkit. Of course, the imported library can be processed further using the functionality provided by ADOxx.

5 CASE STUDY

In the following, an application of MUVIEMOT by means of a case study form the enterprise modelling domain is discussed. For the case study, the Semantic Object Model (SOM) (Ferstl and Sinz, 2013) enterprise modelling method is selected. The SOM method is based on a multi-layer approach, combining the layers *enterprise plan*, *business processes*, and the *specification of resources*. Each layer is defined by one or more interrelated viewpoints. Therefore, SOM is a suitable candidate for the evaluation of MUVIEMOT in general and the transformation algorithm in particular. The case study concentrates on the business process (bp) layer of SOM (Ferstl and Sinz, .2006). The creation of SOM bp modelling utilizes a system-oriented multi-view modelling approach (Bork and Sinz, 2013).

SOM bp models consist of four viewpoints, a structural viewpoint called *Interaction Schema*, a behavioural viewpoint called *Task-Event Schema* and viewpoints on the *Decomposition of Business Objects and Business Transactions*, respectively. All viewpoints are derived by a projection on the integrated SOM bp meta model (see Figure 6). SOM bp modelling follows the metaphor *"of a distributed system, consisting of autonomous and loosely coupled business objects. Business objects are coordinated by means of business transactions towards the fulfillment of common goals"* (Ferstl and Sinz, 2013). The goals for SOM bp modelling are manifold and not limited by the authors of SOM. Initially, SOM was created to enable analysis of already existing and the specification of to-be enterprise systems. All components identified have been modelled in a comprehensive Modelling Scenario model using the MUVIEMOT tool (see Figure 5 for an excerpt of the created MUVIEMOT models).

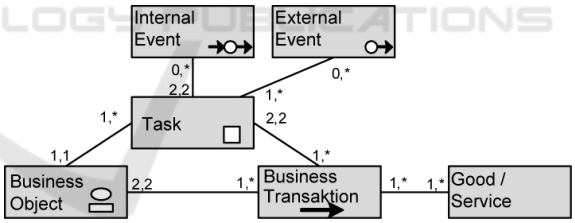

Figure 6: SOM business process meta model (Ferstl and Sinz, 2013).

In order to derive an initial implementation for the SOM method, it was not enough to define the Modelling Scenario. For each meta model (i.e., the SOM bp meta model) and for each viewpoint additional Meta Model models and View models must be modelled, respectively. The SOM bp meta model has been realized in the MUVIEMOT tool by mapping the concepts Business Object, Task, External Event, Internal Event to the ADOxx *Modelling Class*, whereas the concept of a Business Transaction and Object-internal Event have been mapped to the ADOxx *Relation Class* concept (cf. the model in the lower left side of Figure 5).

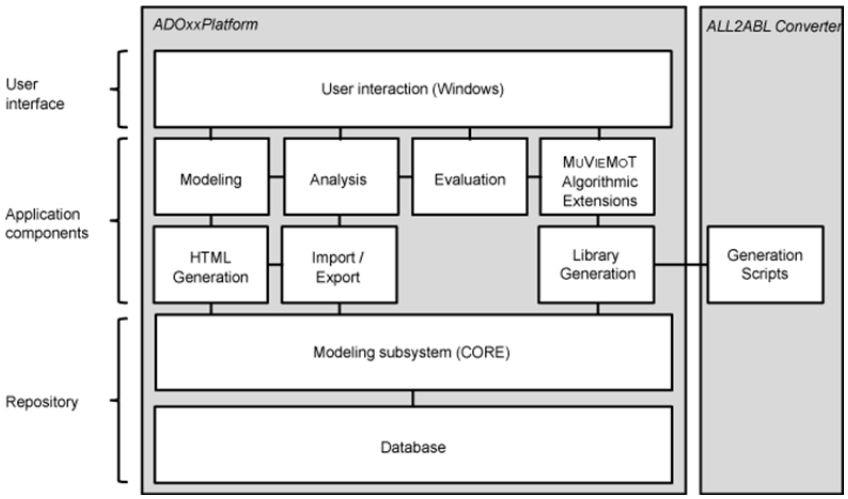

Figure 7: Architecture of the MuVieMoT modelling tool.

Together with attributes describing the elements and constraints considering the combination of them, the meta model of SOM has been completely modelled using the tool. For each of the SOM bp viewpoints, a Viewpoint model has been modelled. Each viewpoint is defined as a projection onto the SOM bp meta model. MuVieMoT provides an easy to use copy & paste functionality for copying elements of the meta model and pasting them into the Viewpoint model. All selected Modelling Class and Relation Class concepts are copied together with their attribute values.

After the generation of the models into over 1700 lines of ALL code, the *ALL2ABL* converter service provided by the BOC can be used to generate an ADOxx application library. This library can then be imported into the ADOxx Development Toolkit enabling the creation of models according to the SOM method.

Due to the experience of implementing a SOM modelling tool on ADOxx from scratch (cf. Bork and Sinz, 2010) a comparison according to the efficiency and the usability of the two development approaches can be done. *MuVieMoT* and the generation of the ALL extremely foster the conception and implementation of multi-view modelling tools on a higher abstraction level in a more user-friendly way. A method engineer, following the approach, can concentrate on the domain-concepts and define the conceptual design of a MVM tool in an intuitive way. He or she doesn't have to go into the technical details e.g., how to define a meta model on the meta modelling platform or how to generate the multiple modelling editors. Due to the automatic transformation, most of the functionality is already generated. A tool developer therefore only has to concentrate on the very specific requirements of a modelling method that cannot be generated or implemented automatically. These specific requirements concentrate on the modelling procedure and mechanisms & algorithms of the method. However, the generated models contribute to the discussions between method engineer and tool developer by utilizing documentation and analysis needs on a high abstraction level.

The case study showed the operability of the approach and the transformation. The increase of efficiency and usability in development encouraged us to develop further functionality for MuVieMoT in order to generate even more code automatically. Consequently, future research will concentrate on two major issues: First, a comprehensive user test should be undertaken in order to evaluate not only the operability but also the efficiency of the approach compared to conventional implementation from scratch. This evaluation should include several different multi-view modelling methods. The divers set of modelling methods available within the Open Models Initiative (OMI) eases the access to more candidates. Second, the model-based definition of view constraints and the transformation of this constraint models into AdoScript code should be emphasized. This would enable the method engineer to define the complex constraints also on a higher abstraction level without considering the specific technical implementation on the platform.

A first prototype of the MuVieMoT modelling tool is being implemented within the Open Models Initiative (Karagiannis et al., 2008) (OMI) at the University of Vienna.

6 CONCLUSION

The paper at hand introduced MUVIEMOT, a set of domain-specific modelling languages for the specification and model-driven development of multi-view modelling tools. The tool is realized using the ADOxx meta modelling platform.

Operability and utility of the approach have been discussed referring to a case study from the enterprise modelling domain.

The current development status of MUVIEMOT limits its functionality to the specification of the modelling language of a multi-view modelling method. Modelling procedure and mechanisms & algorithms are not regarded up to now.

As the results of the case study are very promising, future research will concentrate on broadening the tool support for MUVIEMOT by including the Multi-View Modelling Use Cases and the Conceptual Design phases of the approach.

Recently, researchers have enabled the formal specification of ADOxx meta modelling methods using a mathematical notation, called FDMM (Fill et al., 2012). Introducing the FDMM formalization as another transformation target and/or enlarging the FDMM approach to also formalize the specification of multi-view modelling procedures may contribute to foster the coupling of the MUVIEMOT approach and modelling tool development.

Additionally, coupling MUVIEMOT with the functionality provided by the statistical software environment R would be a very interesting research field (cf. the RUPERT modelling tool (Johannsen and Fill, 2014)).

ACKNOWLEDGEMENTS

The authors would like to thank Hans-Georg Fill, whose expertise and opinions have had a significant influence while writing this paper.

REFERENCES

Alter, S. (2008). Defining information systems as work systems: implications for the IS field. *European Journal of Information Systems*, 17(5):448–469.

Atkinson, C., Gerbig, R., and Tunjic, C. (2013). A Multilevel Modeling Environment for SUM-based Software Engineering. In *Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, VAO '13, pages 2:1–2:9, New York, NY, USA. ACM.

Benington, H. D. (1983). Production of Large Computer Programs. *IEEE Annals of History of Computing*, 5(4):350–361.

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

Bork, D. and Sinz, E. J. (2010). Design of a SOM Business Process Modelling Tool based on the ADOxx Meta-modelling Platform. In de Lara, J., Varro, D., Margaria, T., Padberg, J., and Taentzer, G., editors, *4th International Workshop on Graph Based Tools*, (GraBaTs 2010), Enschede, The Netherlands, pages 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):25–41.

Carlson, C. R. and Arora, A. K. (1979). The updatability of relational views based on functional dependencies. In *Computer Software and Applications Conference, Proceedings. COMPSAC 79*. pages 415–420.

Cicchetti, A., Ciccozzi, F., and Leveque, T. (2011). A hybrid approach for multi-view modeling. *Electronic Communication of the ECEASST*, 50. Dayal, U. and Bernstein, P. A. (1978). On the Updatability of Relational Views., pages 368–377. IEEE Computer Society.

Dayal, U. and Bernstein, P. A. (1982). On the Correct Translation of Update Operations on Relational Views. *ACM Transactions on Database Systems*, 7(3):381–416.

Dijkman, R. M., Quartel, D. A. C., and van Sinderen, M. J. (2008). Consistency in multi-viewpoint design of enterprise information systems. *Information Software Technology*, 50 (7-8):737–752.

Ferstl, O. K. and Sinz, E. J. (2006). Modeling of Business Systems Using SOM. In Bernus, P., Mertins, K., and Schmidt, G., editors, *Handbook on Architectures of Information Systems*, pages 347–367. Springer Berlin Heidelberg.

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

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

Fill, H.-G., Redmond, T., and Karagiannis, D. (2012). FDMM: A Formalism for Describing ADOxx Meta Models and Models. In Maciaszek, L., Cuzzocrea, A., and Cordeiro, J., editors, *14th International Conference on Enterprise Information Systems*, pages 133–144.

Finkelstein, A. and Fuks, H. (1989). Multi-party Specification. *SIGSOFT Software Engineering Notes*, 14(3):185–195.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., and Goedicke, M. (1992). Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, 2.

Frank, U. (1994). *Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung*. Berichte der Gesellschaft für Mathematik und Datenverarbeitung. Oldenbourg, München.

Frank, U. (2010). Outline of a method for designing domain-specific modelling languages. ICB-Research Report 42, University Duisburg-Essen.

Frank, U., Strecker, S., Fettke, P., Brocke, J., Becker, J., and Sinz, E. (2014). The Research Field "Modeling Business Information Systems". *Business & Information Systems Engineering*, 6(1):39–43.

Harel, D. and Rumpe, B. (2000). Modeling Languages: Syntax, Semantics and All That Stuff Part I: The Basic Stuff. Technical report, The Weizmann Institute of Science.

Harel, D. and Rumpe, B. (2004). Meaningful Modeling: What's the Semantics of „Semantics"? *IEEE Computer*, 37(10):64–72.

Johannsen, F. and Fill, H.-G. (2014). RUPERT: A modelling tool for supporting business process improvement initiatives. In *Proceedings of the 2014 International Conference on Design Science Research in Information Systems and Technology, DESRIST'2014*, Miami, USA.

Junginger S, Kühn H, Strobl R, Karagiannis Dimitris (2000) Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. *Wirtschaftsinformatik* 42(5):392–401

Karagiannis, D., Grossmann, W., and Hoefferer, P. (2008). Open Model Initiative: A Feasilbility Study. http: //cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf, last checked: 2014-06-14.

Karagiannis, D. and Kühn, H. (2002). Metamodeling Platforms. In Bauknecht, K., Min Tjoa, A., and Quirchmayr, G., editors, *Third International Conference ECWeb 2002 – Dexa*, 2002, page 182, Aix-en-Provence, France. Springer-Verlag, Berlin, Heidelberg.

Keller. A. M. Algorithms for Translating View Updates to Database Updates for Views Involving Selections, Projections, and Joins. In *Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, PODS '85, pages 154-163, New York, NY, USA, 1985. ACM.

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

Kruchten, P. (1995). Architectural Blueprints - The "4+1" View Model of Software Architecture. *IEEE Software*, 12(6):42–50.

Lechtenbörger, J. (2003). The Impact of the Constant Complement Approach Towards View Updating. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 49–55, New York, NY, USA. (ACM).

Mili, H., Dargham, J., Mili, A., Cherkaoui, O., and Godin, R. View programming for decentralized development of OO programs. In *Technology of Object-Oriented Languages and Systems*, 1999. TOOLS 30, pages 210-221.

Nassar, M. (2003). VUML: a Viewpoint oriented UML Extension. In *Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, pages 373–376.

Romeikat, R., Roser, S., Muellender, P., and Bauer, B. Translation of QVT Relations into QVT Operational Mappings. In *Theory and Practice of Model Transformations* 2008, pages 137–151. Springer.

Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31.

Sprinkle, J., Rumpe, B., Vangheluwe, H., and Karsai, G. (2010). Metamodelling - State of the Art and Research Challenges, In *Proceedings of the 2007 International Dagstuhl conference on Model-based engineering of embedded real-time systems*, pages 57–76. Springer.

Tolvanen, J.-P. (2005). Domain-specific modeling for full code generation. *Methods & Tools*, 13(3):14–23.

Tolvanen, J.-P. and Rossi, M. (2003). MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators. In *Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pages 92–93, New York, USA. (ACM).

BRIEF BIOGRAPHY

Dimitris Karagiannis is head of the research group knowledge engineering at the University of Vienna. His main research interests include knowledge management, modelling methods and meta-modelling. Besides his engagement in national and EU-funded research projects Dimitris Karagiannis is the author of research papers and books on Knowledge Databases, Business Process Management, Workflow-Systems and Knowledge Management. He serves as expert in various international conferences and is presently on the editorial board of Business & Information Systems Engineering (BISE), Enterprise Modelling and

Information Systems Architectures and the Journal of Systems Integration. He is member of IEEE and ACM and is on the executive board of GI as well as on the steering committee of the Austrian Computer Society and its Special Interest Group on IT Governance. Recently he started the Open Model Initiative (www.openmodels.at) in Austria. In 1995 he established the Business Process Management Systems Approach (BPMS), which has been successfully implemented in several industrial and service companies, and is the founder of the European software- and consulting company BOC (http://www.boc-group.com), which implements software tools based on the meta-modelling approach.